Roope Yli-Hukka

LARAVEL-OHJELMISTOKEHYS

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Kesäkuu 2016

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Kesäkuu 2016	**Tekijä/tekijät** Roope Yli-Hukka
Koulutusohjelma Tietotekniikan koulutusohjelma		
Työn nimi LARAVEL-OHJELMISTOKEHYS		
Työn ohjaaja Kauko Kolehmainen		**Sivumäärä** 61
Työelämäohjaaja		

Tämän opinnäytetyön ensisijaisena lähtökohtana oli halu ymmärtää, tutustua ja oppia hyödyntämään PHP-kehystä verkkosivuston suunnittelussa ja toteuttamisessa. Työn olennaisena osana oli myös vertailla eri PHP-kehysten ominaisuuksia ja suosiota ja valita niiden perusteella mieluinen kehys.

Valinnan teon jälkeen oli ensiksi hankittava tarpeellinen tieto kehyksestä ja aloitettava sen käytön opiskelu. Lopuksi testattiin vielä opittuja asioita käytännössä eli luotiin testisivusto sekä vertailtiin koko luomisprosessia kehyksen kanssa vertailukohteena aikaisemmat kokemukset verkkosivujen suunnittelussa ja toteuttamisessa.

Kehysvalintana oli loppujen lopuksi Laravel. Tässä raportissa käsitellään pääasiassa Laravelin ominaisuuksia ja sen käytön hyötyjä. Sen lisäksi raportissa käsitellään myös muita verkkosivujen tuottamiseen liittyviä keskeisiä asioita yleisesti, kuten esim. suunnittelu-, rakenne- ja ulkoasuratkaisuja.

Asiasanat
Kehys, Laravel, PHP

ABSTRACT

Centria University of Applied Sciences	Date June 2016	Author/s Roope Yli-Hukka
Degree programme Information Technology		
Name of thesis LARAVEL SOFTWARE FRAMEWORK		
Instructor Kauko Kolehmainen		**Pages** 61
Supervisor		

The main motive for this thesis work was to learn to understand and use a PHP Software framework in creating a website. A fundamental part of the work was also to compare the features of different frameworks and their popularity to help choose an appropriate framework to use.

After making the choice it was important to acquire the necessary information about the framework and to start studying its use. Finally the learned skills were tested in practice by creating a test website and comparing the whole creation process with the framework to previous experiences in designing and creating websites.

The choice of framework ended up being Laravel. This report deals mainly with Laravel's features and the advantages of its use. The report also addresses other central issues regarding the creation of websites in general; e.g. design, structure and layout solutions.

Asiasanat
Framework, Laravel, PHP

SISÄLLYS

KUVIOT

1 JOHDANTO

Opinnäytetyön päämääränä oli pääasiassa perusteiden oppiminen PHP-ohjelmistokehyksestä ja niiden soveltaminen myös käytännössä eli verkkosivujen tuottamisessa. Tavoitteena oli valita mahdollisimman laadukas, monipuolinen ja nykyaikainen eli melko uusi ohjelmistokehys. Työn taustalla oli myös halu herättää intohimo verkkosivujen tuottamista kohtaan uudelleen oppimalla kaikenlaista uutta. Koska verkkosivujen tuottaminen oli jo valmiiksi niiden harrastamisen kautta tuttua, niin haluttiin myös saada uusia näkemyksiä ja kokeilla mahdollisimman paljon uutta sekä parantaa koko verkkosivujen suunnittelu- ja toteuttamisprosessia.

Aivan aluksi työssä vertailtiin eri vaihtoehtoja suosituista PHP-kehyksistä ja erilaisten kriteerien perusteella valittiin mieleinen. Mainittakoon myös, että alun perin harkittiin myös jonkin sisällönhallintaohjelmiston kuten esim. Drupalin, WordPressin tai Joomlan valitsemista opinnäytetyön pohjaksi, mutta päädyttiin kuitenkin sen sijaan PHP-kehykseen. Sisällönhallintaohjelmiston vahvuutena on usein sen nopeampi oppimiskäyrä ja helppokäyttöisyys verrattuna ohjelmistokehykseen, mutta alustasta haluttiin mahdollisimman monipuolinen ja muokattava ilman niitä rajoitteita, joita sisällönhallintaohjelmistoilla tyypillisesti on.

Lopullisen PHP-ohjelmistokehyksen valinnassa mietittiin ainakin näitä seikkoja: muokattavuus, ominaisuuksien määrä, tietoturva, monipuolisuus, käyttäjäkunnan ja kommuunin aktiivisuus. PHP-ohjelmistokehyksiä on lukemattoman monta, joten heti aluksi jouduttiin tekemään jonkinlainen esikarsinta ja valinta, mitä kehyksiä otettiin tarkempaan vertailuun mukaan. Työssä päädyttiin valitsemaan viisi suosittua ohjelmistokehystä, joiden välillä lopullinen valinta tehtiin. Vaikka suosittu ei välttämättä aina tarkoita samaa kuin paras, kannattaa ottaa huomioon, että suosituimmilla ohjelmistokehyksillä on yleisesti suuremmat käyttäjämäärät, joten kysymyksien tai ongelmatilanteiden syntyessä on helpompi löytää tai kysyä apua ongelmiin, eikä opiskelu ja työnteko jumitu. Jokaista kehystä ei tietenkään millään voitu kokeilla kovin perusteellisesti etukäteen, mutta erilaiset kyselyt ja tilastotiedot mahdollistivat suhteellisen nopean valinnan esivalittujen kehysten väliltä.

Opinnäytetyö jakaantui neljään päävaiheeseen, joista ensimmäisenä oli eri ohjelmistokehys-

ten vertailu ja vertailun perusteella tehtävä valinta. Toisena vaiheena oli hankittava tarpeellinen tieto työympäristöstä sekä siitä, kuinka itse kehyksen käyttöönotto ja käyttö onnistuu. Työympäristön valitsemisen ja kehyksen oppimisen jälkeen haluttiin päästä kokeilemaan kehyksen ominaisuuksia myös käytännössä, eli pienimuotoisen testisivuston muodossa. Testisivuston aiheeksi valittiin verkkokauppa, koska sen ominaisuudet ja vaatimukset ovat melko monipuolisia. Täysin toimivan verkkokaupan rakentaminen kokonaan vain testisivustoa varten olisi kuitenkin ollut aivan liian laaja ja haastava aihe, joten osa verkkokaupan tavallisista toiminnallisuuksista, kuten esimerkiksi maksaminen, rajattiin pois. Työhön kuului myös pienimuotoista testausta osana testisivuston toteuttamista.

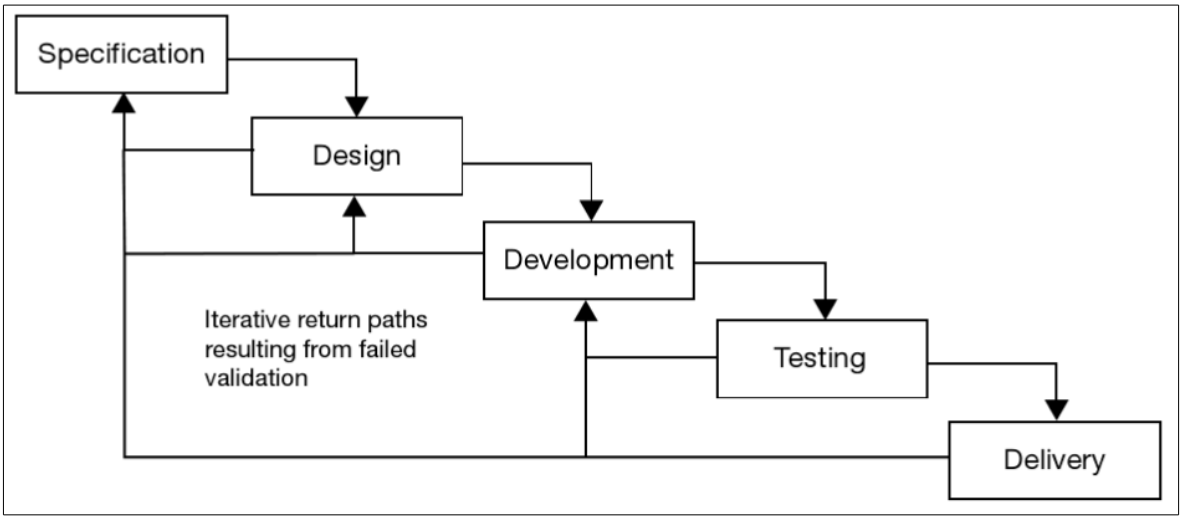
2 OHJELMISTOTUOTANTO

Ohjelmistotuotantoa käytetään yhteisnimityksenä niille työnteon ja työnjohdon menetelmille, joita käytetään tietokoneohjelmista ja tietokoneohjelmistoista. Ohjelmistotuotanto on siis käsitteenä laaja. Ohjelmistotuotanto käsittää oikeastaan kaiken tietokoneohjelmistojen valmistukseen liittyvät seikat, eli kaikki ohjelmaan tai ohjelmistoon liittyvät valmistamismenetelmät ja prosessinhallinnan. (Webopas.net. 2013.) Ohjelmistotuotantoon liittyy kaikenlaisia määrittelyitä, analysointia, suunnittelua, ohjelmointia, testausta ja lopuksi myös monenlaisia toimittamiseen ja ylläpitoon liittyviä asioita. Myös projektinhallinta on iso osa ohjelmistotuotantoa. (Rouse 2010.)

2.1 Ohjelmistoprojekti

Yleinen tapa lähestyä ohjelmistoprojektia on käyttää SDLC-mallia (Engl. Software Development Life Cycle), eli ohjelmistokehityksen elinkaarimallia. Malleja on monenlaisia ja niissä on jotain eroja, mutta yleensä ne muistuttavat paljon toisiaan. Elinkaarimallien valintaan vaikuttavat monet seikat, kuten projektin koko tai laajuus. Yleisiä malleja ovat ainakin vesiputous (Engl. Waterfall), ketterä (Engl. Agile), V-malli (V-Shaped model) sekä SDM eli spiraalimetodi (Engl. Spiral method). (Lecky-Thompson 2005.)

Otetaan esimerkiksi vesiputousmalli. Vesiputousmallissa on 5 päävaihetta: määrittely (Engl. specification), suunnittelu (Engl. Design), kehitys (Engl. Development), testaus (Engl. testing) ja lopuksi toimitus (Engl. delivery). Kaikki kohdat jaetaan usein vielä pienempiin osiin, eli esimerkiksi määrittelyssä voidaan määritellä aluksi vaikka toiminnallisia ominaisuuksia ja toisena osana ei-toiminnallisia ominaisuuksia. Toiminnallinen ominaisuus voi olla esimerkiksi jokin toteutettava ominaisuus, kuten kirjautuminen tai maksaminen. Ei-toiminnallinen ominaisuus taas voi liittyä sovelluksen nopeuteen, vakauteen tai vaikka tietoturvaan. Tässä työssä ei hyödynnetty suoraan mitään tarjolla olevaa mallia, vaan ainoastaan tiedostettiin niiden olemassaolo ja sovellettiin niiden periaatteita. (Lecky-Thompson 2005.) Kuviossa 1 on esitettynä yksinkertaisen vesiputousmallin toimintaperiaate.



KUVIO 1: Vesiputousmalli (mukaillen Lecky-Thompson 2005)

2.2 MVC-arkkitehtuuri

MVC on yleisesti käytetty suunnittelumalli ohjelmistokehityksessä, ja se on kaikista yleisin käytetyistä arkkitehtuureista varsinkin PHP-ohjelmistokehyksissä, missä itse koodi on jaettu kolmeen pääosaan: malleihin (model), näkymiin (view) sekä kontrollereihin (controller). MVC-arkkitehtuurin nimi tulee siis vain suoraan sen rakenteesta, eli jokainen kirjain viittaa yhteen sen pääosaan. MVC-mallin eri osien tarkemmat käyttötarkoitukset vaihtelevat hiukan toteutuksien mukaan, koska eri ohjelmistokehysten välillä on hiukan vaihtelua. (Bari & Syam 2008, 8.) Vaikka MVC:n pääidea on aina sama, siitäkin on hiukan eri versioita, eli eri MVC-mallit voivat poiketa toisistaan hiukan.

Yleinen ongelma ohjelmoidessa www-ympäristössä on, miten erottaa käyttöliittymän (näkymän) erilaiset luokat, joita hoitavat monimutkaisen ajattelun (malli) ja muun ohjelman sisäisen prosessoinnin ja päätöksenteon (controller) siten, että ne edustavat kolmea erillistä, erotettavissa olevaa komponenttia. Yleisesti käytetty MVC-malli tarjoaa ratkaisun juuri tähän kyseiseen ongelmaan. (Lecky-Thompson & Nowicki 2009, 306.)

2.2.1 Malli (Model)

Malli on sovelluksen sydän. Yleisesti ottaen se viittaa luokkien ryhmään, jotka on kehitetty käsittelemään eri prosesseja sovelluksesi käyttäytymisessä. Se huolehtii tiedon noutamisesta sekä manipuloi tietoja, joita käyttäjä on syöttänyt. Sillä on suora yhteys kontrolleriin eikä sillä ole mitään näkyvyyttä näkymässä. Kontrolleri toimittaa mallin näkymälle samalla antaen käyttöohjeet, ja näkymä manipuloi vastaanotettua tietoa ja muuttaa sen ihmisen käytettävissä olevaan ulostuloon. (Lecky-Thompson & Nowicki 2009, 303–307.)

2.2.2 Näkymä (View)

Näkymä on nimensä mukaisesti näkymä, jonka käyttäjä saa pyytäessään kontrollerilta tietoa. Kontrolleri voi päättää, mitä tietoja käyttäjän tarvitsee nähdä tai tietää. Käyttäjä kommunikoi kontrollerin kanssa esimerkiksi lomakkeiden ja linkkien avulla. Oikein toteutetussa näkymässä ei pidä olla suoria viittauksia luokkiin eikä ehdottomasti mitään kommunikointia tietokantoihin. (Lecky-Thompson & Nowicki 2009, 307.)

2.2.3 Kontrolleri (Controller)

Kontrolleri on kerros näkymän ja mallin välillä. Kontrolleri päättää, mitä tehdä jokaisen pyynnön perusteella. Päätös perustuu moniin asioihin, kuten vaikkapa käyttäjän nykyinen sijainti sivustolla, käyttäjän toiminto sivulla ja sivun tila (esim. lomakkeen arvot ja muut erityistilat, kuten kirjautumisen tila). Kontrolleri kommunikoi mallin kanssa tarvittaessa ennen kuin se ohjaa käyttäjän uuteen näkymään. (Lecky-Thompson & Nowicki 2009, 307.)

2.3 Olio-ohjelmointi (OOP)

Olio-ohjelmointi eli OOP (engl. Object-oriented programming) on eräänlainen ohjelmoinnin lähestymistapa, jonka avulla on helpompi mallintaa koodi vastaamaan todellisen maailman teh-

täviä, prosesseja ja ideoita jotka sovellus on suunniteltu käsittelemään. Olio ohjelmointi vaatii tavallisesta ohjelmoinnista poikkeavan ajattelutavan ohjelmistoja rakentaessa. Olio-ohjelmoinnin lähestymistapa mahdollistaa sovelluksen mallintamisen joukkona yhteistyötä tekeviä objekteja, eli olioita jotka käsittelevät itsenäisesti tiettyjä toimintoja.

Esitetään havainnollistava analogia, jossa rakennetaan taloa. Putkimiehet hoitavat putket ja sähkömiehet hoitavat johtoja. Putkimiesten ei tarvitse tietää, onko makuuhuoneen virtapiiri kymmenen amppeeria vai kaksikymmentä ampeeria. Heidän pitää huolehtia ainoastaan omista aktiviteeteistaan. Pääurakoitsija varmistaa, että kaikki aliurakoitsijat hoitavat omat työnsä, mutta ei ole välttämättä kinostunut jokaisen työtehtävän yksityiskohdista. Olio-ohjelmoinnin lähestymistapa on samankaltainen siinä mielessä, että jokainen olio piilottaa sen täytäntöönpanojen yksityiskohdat muilta olioilta. Systeemin muille komponenteille on yhdentekevää, miten se tekee työnsä. Vain olion toimittamalla palvelulla on väliä. Luokkien ja olioiden käsitteet ja tavat, jolla näitä ideoita voi hyödyntää ohjelmistoja kehittäessä, ovat olio-ohjelmoinnin perusideoita. Tämä menettely on tavallaan proseduraalisen ohjelmoinnin vastakohta, joka siis on ohjelmointia funktioita ja tietorakenteita käyttäen. Ohjelmointikielistä esim. C- tai Pascal-kieli ovat tyypillisiä proseduraalisia ohjelmointikieliä. Vastavaasti taas olio-ohjelmointia tukevia ohjelmointikieliä ovat muun muassa Java, C# ja C++. Aikaisemmat PHP-versiot eivät myöskään tukeneet olio-ohjelmointia, mutta sillekin saatiin tuki PHP5-version mukana ja se sai myöhemmin vielä joitain parannuksia. (Lecky-Thompson & Nowicki 2009, 3–4.)

OOP on myös keskeisessä osassa Laravel-ohjelmistokehyksessä, eli olisi suositeltava ainakin ymmärtää edes hiukan olio-ohjelmoinnin perusteita ennen kuin aloittaa Laravelin opiskelun. Laravel-opetusvideoita tarjoava Laracasts-sivusto tarjoaakin nopean tavan oppia olio-ohjelmoinnin perusteita hyvillä esimerkeillä Laravel ympäristössä videosarjassa nimeltä ”Object-Oriented bootcamp”. (Way 2014.)

3 WEB-SOVELLUSTEN KEHITTÄMINEN

Ennen web ohjelmistokehittäjien piti ajatella nykyään toissijaisia asioita, kuten esim. sivuston lataamisnopeutta ja eri selaimille toteuttamista. Nämä ovat melko mitättömiä ongelmia nykyajan web ohjelmistokehittäjälle. Kaikki web sovellusten kehittämiseen liittyvät seikat ovat kehittyneet lähiaikoina uskomattoman nopeasti. Monesta asiasta on tullut paljon monipuolisempia ja usein samalla monimutkaisempia, eli voidaan jopa päätellä, että erikoistumiselle on enemmän tarvetta mitä ennen. Hakukoneoptimointi on täydellinen esimerkki. Kilpailun lisääntyminen ja kokoajan muuttuvat vaatimukset pakottavat suunnittelijat erikoistumaan, eli sivuston suunnittelijoiden pitää olla ajan tasalla muutoksista ja keksiä luovia ratkaisuja uusien käyttäjien löytämiseksi ja houkuttelemiseksi. Myös sivuston suunnittelemisessa vaaditaan suunnittelijoilta paljon enemmän kuin ennen. Toisaalta voidaan myös todeta, että usein sivustojen tekemisessä ei oteta enää samalla tavalla riskejä eikä kokeilla kovin paljon uutta, eli suurilta osin ollaan ikään kuin jämähdetty paikoilleen, vaikka poikkeuksiakin toki on. Valtaosa sivustoista on siis sellaisia, jotka on jo todettu aikaisemmin toimiviksi ratkaisuiksi. Osasyynä tälle voi myös olla, että toisenlaiset ratkaisut vaativat myös enemmän osaamista. (Echher 2014, 1.)

3.1 Web-suunnittelu

Web sovellusten kehitystä kutsutaan usein nimellä web suunnittelu, eli englanniksi web design. Suunnittelijasta käytetään sen englanninkielistä nimitystä web designer usein myös suomessa. Web suunnittelu on melko monimuotoinen käsite, koska ammattilaiset määrittelevät sen koko ajan eri tavoin. Joku voi määritellä sen sivuston back-end-toiminnallisuuksien ohjelmoinniksi, kun taas joku toinen määrittelisi sen front-end ulkoasun suunnitteluksi, joka antaa kuvan yrityksestä tai yksilöstä, jota sivusto edustaa. Todellisuudessa molemmat määrittelyt ovat oikein. Osaavan web suunnittelijan täytyy ymmärtää monet tekniset ja taiteelliset web designin osat, vaikka molempiin samanaikainen erikoistuminen ei ole mitenkään välttämätöntä. Monipuolisuus on usein suuri valtti, mutta erikoistumistakin tarvitaan. Nykyaikana monien tapauksien teknisiin standardeihin liittyy usein dynaamisia tietokantoihin perustuvia

sivustoja, jotka ovat monipuolisia, skaalautuvia, tehokkaita ja hakukoneystävällisiä. Jos nämä sivustot koostuvat kuitenkin vain muotoilemattomista sivuista, joissa on vain mustaa tekstiä valkoisella pohjalla, ne eivät saa suuren yleisön suosiota kovinkaan helpolla. Toisaalta, jos sivusto käyttää uusimpia graafisen suunnittelun menetelmiä, mutta sisältää vain staattisia sivuja ilman mitään toiminnallisuutta, sivusto tulee olemaan epäkäytännöllinen käyttää. Monipuolinen osaaminen on siis suositeltavaa myös tässä asiassa. (Echher 2014, 1–2.)

3.1.1 Asiakaspuoli (Front-end)

Kun puhutaan frontendin eli asiakaspuolen osasta sovelluksesta, niin todellisuudessa puhutaan siitä osasta, joka on nähtävissä ja jonka kanssa voidaan olla vuorovaikutuksessa jollain tavoin. Frontend koostuu yleensä kahdesta osasta, eli web sovelluksen suunnittelusta (Engl. web design) ja sen asiakaspuolen kehittämisestä (Engl. frontend web development). Aikaisemmin, kun keskusteltiin web kehityksestä, viitattiin yleensä backend- eli palvelinpuolen ohjelmointiin, mutta viime vuosina on ollut todellinen tarve erottaa toisistaan graafiset suunnittelijat ja sellaiset suunnittelijat, jotka kirjoittavat HTML- ja CSS-koodia. Myöhemmin rinnalle tuli myös JavaScript ja jQuery, eli myös siihen erikoistuneet suunnittelijat pystyttiin rajaamaan vastaavasti. Yleensä kuitenkin HTML- ja CSS-koodia kirjoittavien oletetaan osaavan myös JavaScriptiä ja jQueryä, joten heidät voidaan laittaa samaan käsitteeseen.

Eli kun puhutaan web suunnittelijasta, pitäisi vielä määrittää, onko kyseessä graafinen suunnittelija, joka käyttää esim. Photoshopia tai Fireworkia ulkoasun luomiseen, vai tarkoitetaanko ohjelmoijaa, joka kirjoittaa asiakaspuolen koodia, eli HTML-, CSS-, JavaScript- ja jQuery-koodia. Asiakaspuolen suunnitteluun kuuluvat siis kaikki näkymiin vaikuttavat koodit, fontit, kuvat, linkit, pudotusvalikot, liukusäätimet, lomakkeet jne., eli kaikki frontend-elementit. Toiminnallisuuden lisäämiseksi tarvitaan kuitenkin backend- eli palvelinpuolen teknologioita. (Long 2012.)

3.1.2 Palvelinpuoli (Back-end)

Palvelinpuolen suunnittelua kutsutaan yleisesti backend-suunnitteluksi. Backend koostuu yleensä kolmesta osasta: palvelin, sovellus ja tietokanta. Otetaan esimerkiksi sivusto, jossa varata lippuja. Jos varaa lennon tai ostaa lippuja, avaa yleensä sivun, jossa on vuorovaikutuksissa frontend-osan kanssa. Kun on syöttänyt tiedon lomakkeeseen, tieto syötetään tietokantaan, joka sijaitsee palvelimella. Kaikki tieto pysyy palvelimella, eli kun kirjautuu takaisin sovellukseen tulostamaan lippuja, kaikki aiemmin syötetty tieto on edelleen käyttäjätilillä tietokannan ansiosta.

Henkilöä, joka rakentaa kaikki nämä teknologiat toimimaan yhdessä, kutsutaan nimellä backend-kehittäjä. Yleisimpiä backend-ohjelmointikieliä ovat ainakin PHP, Ruby ja Python. JavaScriptiä käytetään myös backend-ohjelmointiin, mutta sitä käytetään myös frontend-ohjelmointiin. Erinlaiset alusta- ja kehysratkaisut, kuten esim. Laravel auttavat backend-suunnittelijaa toteuttamaan monimutkaisiakin sovelluksia helpommin ja nopeammin. (Long 2012.)

3.1.3 Layout

Layout on yleisesti käytetty käsite web suunnittelussa sivuston graafisesta ulkoasun suunnitelmasta. Layout on siis eräänlainen sivuston pohjapiirros. Myös layoutin tekemisessä on usein monta eri vaihetta, kuten vaikka aivan aluksi suuntaa antava piirros paperille. Lopullinen layout yleensä tehdään jollakin siihen soveltuvalla ohjelmalla. Yleensä se tehdään millä tahansa saatavalla olevalla grafiikankäsittelyohjelmalla, kuten vaikka Photoshopilla. Tällä tavoin on helppoa sijoittaa asiat paikoilleen ilman, että tarvitsee huolehtia tyylitiedostoista tai muusta koodista. Layoutin avulla on myös helppo testata monenlaisia asioita, kuten vaikka eri värimaailmoja tai millä tavoin jokin tietty fontti sopii. Koko sivuston front-end-koodaaminen on paljon hitaampaa kuin layoutin tekeminen, joten layout tehdään juuri sen takia, että saadaan jonkinlainen kuva, miltä sivusto tulee loppujen lopuksi näyttämään. Alkuperäiseen layoutiin on myös helppo tehdä muutoksia myös keskellä projektia, ja sitä on helppo jakaa mahdollisesti muille projektissa toimiville henkilöille. Tällä tavoin eri tehtäviä ja ominaisuuk-

sien toteuttamista on helppo havainnollistaa projektissa työskenteleville. Web-projekteissa layout voi olla hyvinkin tärkeä työvaihe ja suuri osa lopullista tuotosta, vaikka siihen ei ohjelmointiosaamista vaaditakaan. Tyylitiedoston luomisvaiheessa layoutin olemassaolo helpottaa myös lopullisen tyylitiedoston luomisessa, varsinkin jos projektin layoutin tekijä ja tyylitiedoston ohjelmoija eivät ole yksi ja sama henkilö.

3.1.3 Hakukoneoptimointi (SEO)

SEO (Search Engine Optimization) eli hakukoneoptimoinnilla tarkoitetaan erilaisia toimenpiteitä, joilla saadaan näkyvyyttä hakukoneissa. Hakukone sijoittaa haettujen sanojen perusteella tuloksia, jotka sopivat parhaiten haettuun asiaan tai asioihin. Eli hyvin sijoittuvat tulokset näkyvät heti haun yläosassa ja laskevat sijoituksen mukaan. Korkeita sijoituksia voi myös ostaa suoraan hakukoneyhtiöiltä, mutta esim. Google-haussa nuo hakutulokset näkyvät hiukan normaaleista hakutuloksista poikkeavina. Sijoitukset määrittyvät sivujen tai dokumenttien sisällön perusteella, mutta niihin voi vaikuttaa myös muilla tavoin, kuten description-meta-elementtiä käyttäen eli antamalla sivulle vapaasanaisen kuvauksen. Kuvaukseen kannattaa myös sijoittaa avainsanoja, joita hakukone voi käyttää hyödykseen. Myös sivun title eli otsikko on isossa osassa hakukoneoptimointia. (Raittila 2016.)

Hakukoneoptimointi on hyvin tärkeä myös verkkokauppojen suunnittelussa, jotta ihmiset löytäisivät sivuston hakukoneiden avulla. Tässä työssä ei kuitenkaan ollut tarvetta hyödyntää hakukoneoptimointia, koska luotu testisivusto oli tarkoitettu vain Laravelin testaamista varten, eikä esimerkiksi yritystä varten tehtyä kokonaista verkkokauppaa toteutettu.

3.2 Toteutustekniikat

Web sovellusten toteuttamiseen käytetään monenlaisia työkaluja, kuten eri ohjelmointikieliä. Kaikista tärkein ohjelmointikieli web ympäristössä on ehdottomasti HTML, koska se loi perustan www-sivujen luomista varten. HTML ei kuitenkaan ole yksinään kovin käyttökelpoinen oh-

jelmointikieli muihin kuin yksinkertaisiin sivuihin, eli se tarvitsee rinnalle muita kieliä avuksi. Dynaamisten sivujen toteuttaminen vaatii yleisesti myös relaatiotietokantoja kuten esim. SQL-tietokantoja. Tässä osiossa tarkastellaan näitä eri toteutustekniikoita, kuten eri ohjelmointikieliä.

3.2.1 HTML

HTML on ohjelmointikieli, joka kehitettiin verkkosivujen luomista varten. HTML (Hypertext Markup Language) eli suomennettuna hypertekstin merkintäkieli on yksi tärkeimmistä World Wide Web Consortiumin eli W3C:n ylläpitämistä standardeista. W3C on 1994 perustettu kansainvälinen yritysten ja yhteisöjen yhteenliittymä, joka vastaa HTML:n lisäksi muunmuassa myös CSS ja JavaScriptin ylläpitämisestä ja kehittämisestä. (W3C. 2016.)

HTML koostuu pääasiassa joukosta tägejä, joiden avulla selain tulkkaa HTML-tiedoston lopulliseen näkyvään muotoon. Esimerkiksi ””-tägi kertoo selaimelle, että kaikki sitä seuraava teksti halutaan tulostaa paksunnettuna niin kauan, kunnes tägi suljetaan vastaavalla tägillä, jossa on lisänä vinoviiva esim. ””.

Koska HTML-kieli on vanha, siinä on paljon puutteita. Sen puutteita kuitenkin paikataan parin vuoden välein ja sen lisäksi on tarjolla paljon sitä avustavia ohjelmointikieliä, jotka mahdollistavat paljon. HTML:ää avustavia kieliä ovat ainakin JavaScript, CSS ja PHP. (Shannon 2012.)

3.2.2 PHP

PHP on suosittu yleiskäyttöinen skriptikieli, joka sopii erityisesti web-sovellusten kehittämiseen. (php.net) PHP on rekursiivinen akronyymi sanoista ”PHP: hyperteksti prosessori” (Engl. PHP: Hypertext Processor), vaikka alun perin se tarkoitti vain henkilökohtaista kotisivua (Engl. Personal Home Page). (The PHP Group, 2016.) Se on palvelinpuolen skriptikieli ja tehokas työkalu dynaamisten ja interaktiivisten verkkosivujen tekemiseen. Sen avulla voidaan

esim. lukea tai vaikka poistaa tiedostoja, hallinnoida evästeitä tai salata salasana (Engl. Encrypt). (W3Schools.com. 2016.)

3.2.3 CSS

CSS eli Cascading Style Sheets tarkoittaa suoraan käännettynä porrastettuja tyyliarkkeja. Se on alun perin kehitetty WWW-dokumenttien tyyliohjeeksi, eli sen avulla voidaan siis vaikuttaa laajasti sivuston ulkoasuun, mutta nykyään sitä voidaan käyttää myös joissain muissa ohjelmissa. Yksinkertaisimmillaan tyylitiedoston avulla voidaan määrittää vaikka sivuston taustaväri tai fontin koko, mutta sillä on myös jopa todella paljon monimutkaisempia ominaisuuksia, kuten eri objektien varjostus tai objektien dynaamisia ominaisuuksia. World Wide Web Consortium (W3C) on HTML-kielen lisäksi myös CSS-kielen kehittäjä, eli niillä on vahva yhteys toisiinsa.

Ennen tyylitiedostoja sivuston ulkonäköä oli hitaampi ja vaikeampi muuttaa, koska kaikki tyyliin liittyvät seikat tehtiin suoraan HTML-tiedostoon. Jos haluttiin vaikkapa vaihtaa otsikkotekstin väriä, jouduttiin väri asettamaan otsikkotekstin attribuutteihin yksi kerrallaan. Tyylitiedostot säästävät siis paljon sellaista turhaa työtä, joka oli tyypillistä www-ohjelmoinnin aikaisissa vaiheissa, koska yhtä tiedostoa muuttamalla pystytään muuttamaan koko sivuston ulkoasua. Tyylitiedostojen avulla sivusto voi kuitenkin tarvittaessa käyttää myös montaa eri tyylitiedostoa eri tarpeisiin. Esimerkiksi sivun tulostukseen voidaan haluta yksinkertaisempi tiedon esittämistapa, eli siinä tapauksessa voitaisiin käyttää erillistä tyylitiedostoa sivun tulostamista varten. Myös eri laitteille voidaan määrittää omat käytettävät tyylitiedostot. Laitteen pieni näyttö tai resoluutio ovat yleisin syy, miksi erillistä tyylitiedostoa tarvitaan, tai miksi se ainakin parantaa käytettävyyttä ja sitä kautta myös käyttäjäkokemusta. Tyylitiedostojen laatiminen on tärkeä osa www-sivuston front-end ohjelmointia. (Lie 2005; HTML.net. 2015; W3C. 2016.)

3.2.4 Javascript

JavaScript on maailman suosituin ohjelmointikieli. (W3Schools.com. 2016a.) Se on järjestel-

mästä riippumaton olio-skriptikieli. JavaScript on myös kevyt ja pieni ohjelmointikieli. Se voidaan liittää sen isäntäympäristön kuten, web-selaimen olioihin, jonka kautta se voi ohjata niitä ohjelmallisesti. JavaScript on todella monipuolinen ohjelmointikieli, joten sitä käytetään niin front-end kuin myös back-end ohjelmoinnissakin. Myös JavaScriptin kehittäjänä toimii World Wide Web Consortium (W3C). Myös tässä työssä hyödynnettiin hiukan JavaScript-kieltä testisivustoa tehtäessä, mutta ei kuitenkaan suuressa osassa. (MDN. 2016.)

3.2.5 Tietokannat

Käytännössä jokainen vakavasti otettava websovellus käyttää jonkinlaista tietojen varastointimekanismia. Tämmöistä tietovarastoa käytetään sovelluksen asetusten, käyttäjärekisteriin, dynaamisen sisällön varastoimista varten. Yksinkertaistamisen vuoksi tietokantaa voidaan kuvata jättimäisenä Excel-taulukkona, joka on varastoituna jossain päin maailmaa. Eli esimerkiksi käyttäjien yksityiset tiedot ja salasanat ovat yleensä varastoitu johonkin tietokantaan. Tietokantojen tiedot ovat yleensä näkymättömissä käyttäjille ja sovellus päättää, mitä tietoja käyttäjälle näytetään. Tällaista tietoa voisi periaatteessa varastoida yksinkertaisesti vaikka tavalliseen tekstitiedostoon, mutta siitä syntyisi heti ongelmia esim. tietoturvan kanssa, eli tällainen ratkaisu ei ole koskaan suositeltavaa, vaan tieto kannattaa varastoida turvallisesti esimerkiksi SQL-tietokantaan. (W3Schools.com. 2016d; Lecky-Thompson & Nowicki 2009, 127; Heng 2015; Ntchosting.com. 2016; Long 2012.)

SQL (Engl. Structured Query Language) on standardikieli tietokantojen hallinnoimiseen. Usein tietokantoihin yhdistetään jonkinlainen tietokantojen hallintajärjestelmä eli DBMS (Engl. Database management system). Yleisesti käytetyt tietokannat ovat relaatiotietokantoja, eli myös SQL on relaatiotietokanta. Tunnettuja tietokantojen hallintajärjestelmiä ovat ainakin MySQL, PostgreSQL, Oracle, SQlite, Microsoft SQL Server ja Sybase. Tämän opinnäytetyön testisivustossa käytettiin SQlite pohjaista tietokantaa, koska sitä suositeltiin pienemmille projekteille Laracast-sivuston opetusvideoissa. (W3Schools.com. 2016d; Lecky-Thompson & Nowicki 2009, 127; Heng 2015; Ntchosting.com. 2016; Long 2012.)

4 SISÄLLÖNHALLINTAJÄRJESTELMÄT

Sisällönhallintajärjestelmällä ja ohjelmistokehyksellä on paljon yhteistä, ja ne onkin helppo sekoittaa keskenään, koska niiden välinen raja voi olla joissain tapauksissa hyvinkin häilyvä. Sisällönhallintajärjestelmä tai kehys voi olla myös niin sanotusti näiden kahden hybridi, eli siinä voi olla molempien parhaita ominaisuuksia. Tällaista hybridiä kutsutaan laajasti sisällönhallintakehykseksi (Eng. Content Management Framework, eli lyhennyksenä CMF). Sisällönhallintakehyksen esimerkkinä voidaan ottaa Drupal, joka on näistä kaikista tunnetuin. Drupal ei kuitenkaan aina ole ollut sisällönhallintakehys, vaan se on pikku hiljaa alkanut muistuttaa enemmän ja enemmän ohjelmistokehystä vasta sen 4.7-version myötä. Nykyään Drupalin ominaisuuksiin kuuluu muun muassa MVC-arkkitehtuurin tukeminen, jota pidetäänkin yhtenä tärkeimmistä PHP-ohjelmistokehysten ominaisuuksista. Täsmennettäköön siis, että MVC-arkkituhtuuri ei ole yleisesti tuettu ominaisuus sisällönhallintajärjestelmissä, mutta ohjelmistokehyksissä se on lähes vaadittu ominaisuus.

Ohjelmistokehyksen ja sisällönhallintajärjestelmän erottavat lisäksi ainakin tapa, miten ne on eri alustoissa hoidettu sekä neljä seuraavaa tärkeää ominaisuutta: tietoturva, joustavuus, päivitykset ja käyttäjäkokemus. Tässä luvussa käydään läpi näitä avainseikkoja samalla vertaillen alustan valitsemisen helpottamiseksi.

4.1 Tietoturva

Hyvä tietoturva on varsinkin nykyään hyvin ajankohtainen ja tärkeä www-sivujen kehitystyössä. Tietokantojen mahdollinen haavoittuvuus on ehkä tärkein seikka, joka kannattaa ottaa huomioon sivustoa rakentaessa. Sivustolla voi kuitenkin löytyä myös muita ongelmia tai haavoittuvuuksia miltä kannattaa varautua jo ennalta mahdollisimman hyvin. Useimmat sisällönhallintajärjestelmät ovat vapaan lähdekoodin edustajia, joten kaikki koodipohja on vapaasti ladattavissa ja täten myös helposti väärinkäytettävissä. Useat internet kommuunit tuottavat erillisiä lisäosia, widgettejä ja moduuleita sisällönhallintajärjestelmille. Kehittäjät etsivät ja kor-

jaavat haavoittuvuuksia uudemmissa versioissa turvallisemmalla koodilla. Sisällönhallintajärjestelmää käyttäessä on kuitenkin tärkeä yrittää käyttää luotettavia lähteitä lisäosia asentaessa. Vertailussa sisällönhallintajärjestelmään ohjelmistokehyksessä tietoturva on tehokkaampaa. Koska ohjelmistokehyksen koodi on mukautetumpaa, murtautuminen on paljon hankalampaa. Useimpiin kehyksiin on myös sisäänrakennettu salausfunktioita ja suojauksia useimpia haavoittuvuuksia vastaan. Tästä voidaan siis päätellä, että hyvin ohjelmistokehyksellä rakennettu sivusto on turvallisempi sisällönhallintajärjestelmällä tehty kokonaisuus. (Surya 2015.)

4.2 Joustavuus

Ohjelmistoprojektien erilaisten tarpeiden mukaan sivusto voi vaatia monenlaisia funktioita ja integraatioita kolmannen osapuolen sovellusten kanssa. Joskus sivuston täytyy pystyä suorittamaan itsenäisiä operaatioita, jotka yhdistyvät moniin erillisiin järjestelmiin. Sisällönhallintajärjestelmän sovelluksissa voi olla paljonkin funktioita, mutta nuo sovellukset eivät kuitenkaan ole kauhean joustavia. Koska sovellukset on tehty ennakkoon, luovuus voi olla ajoittain rajoitettua. Ohjelmistokehys sisältää paljon kirjastofunktioita ja on myös helposti mukautettava tarpeen mukaan. Eli tästä voidaan päätellä, että ohjelmistokehyksen myös joustavuus ja muokattavuus ovat sisällönhallintajärjestelmää edellä. (Surya 2015.)

4.3 Päivitykset

Päivitysten tärkeyttä ei koskaan voi korostaa liikaa – päivitykset parantavat sivuston tietoturvaa ja usein myös käytettävyyttä uusien ominaisuuksien kautta. Siksi päivitykset kannattaakin ottaa molemmissa tapauksissa käyttöön heti, kun ne ovat saatavilla, vaikka sivusto olisi käyttökelpoinen myös ilman päivityksiäkin. Useimmille sisällönhallintajärjestelmille uusia päivityksiä on saatavilla jopa kuukausittain, kun taas ohjelmistokehyksissä päivitykset ovat usein harvinaisempia ja päivityksien välillä voi usein olla jopa yli puoli vuotta ilman, että tietoturva tai käyttökokemus heikkenevät huomattavasti. Päivitysten suhteen kannattaakin siis ottaa huomioon, kuinka usein sivuston omistaja pystyy näitä päivityksiä hoitamaan. (Surya, S. 2015.)

Päivityksissä kannattaa myös ottaa huomioon muutokset, jotka voivat vaikuttaa sivustojen toiminnallisuuteen ja sivusto voi pahimmassa tapauksessa lopettaa toimimasta kokonaan. Esimerkiksi tiedostorakenteen muutokset voivat hajottaa sivun väliaikaisesti, eli sivusto ei toimi ennen kuin tarvittavat muutokset on tehty. Tällaisia muutoksia tulee päivitysten yhteydessä, jolloin vanha koodi ei välttämättä toimi käytetyn alustan uudemmassa versiossa ilman muutoksia tai vastaavasti uusi koodi ei toimi saman alustan vanhemmassa versiossa.

4.4 Käyttäjäkokemus

Käyttäjäkokemus on ehdottomasti yksi tärkeimpiä seikkoja, joita verkkosivujen suunnittelemisessa ja toteuttamisessa on otettava huomioon. Sisällönhallintajärjestelmät on rakennettu erityisesti verkkosivujen ylläpitäjää huomioiden, kun taas kehysratkaisu ei ole lähelläkään yhtä käyttäjäystävällinen. Kehittäjällä on kuitenkin mahdollisuus rakentaa oma käyttöliittymä saatavilla olevien kirjastofunktioiden avustamana. Sivustosta voi tehdä kuitenkin hyvinkin käyttäjäystävällisen joko kokonaan omalla koodilla tai CSS-kehyksen kuten esim. Bootstrapin avulla. Tämä voi kuitenkin vaatia jopa yllättävänkin paljon ylimääräisiä resursseja. Jo oletuksenakin hyvin käyttäjäystävällinen käyttöliittymä on ehdottomasti sisällönhallintajärjestelmän suurin vahvuus jos sitä verrataan ohjelmistokehyksen kanssa. (Surya, S. 2015.)

4.5 Johtopäätökset

Toimivan alustan valitsemisessa yritykselle on siis kiinnitettävä huomio moneen eri seikkaan ja sen lisäksi on otettava huomioon myös asiakkaan toiveet ja sivuston tarpeet. On siis hyvä ymmärtää molempien, sekä ohjelmistokehyksen, että sisällönhallintajärjestelmän vahvuuksia ja heikkouksia alustan valitsemisen yhteydessä. Varsinkin työelämässä asiantuntemus sisällönhallintajärjestelmistä ja ohjelmistokehyksistä on ehdottomasti suositeltavaa hankkia, jotta jokaisen projektin alustaksi löytyisi aina paras mahdollinen vaihtoehto.

5 PHP-OHJELMISTOKEHYKSET

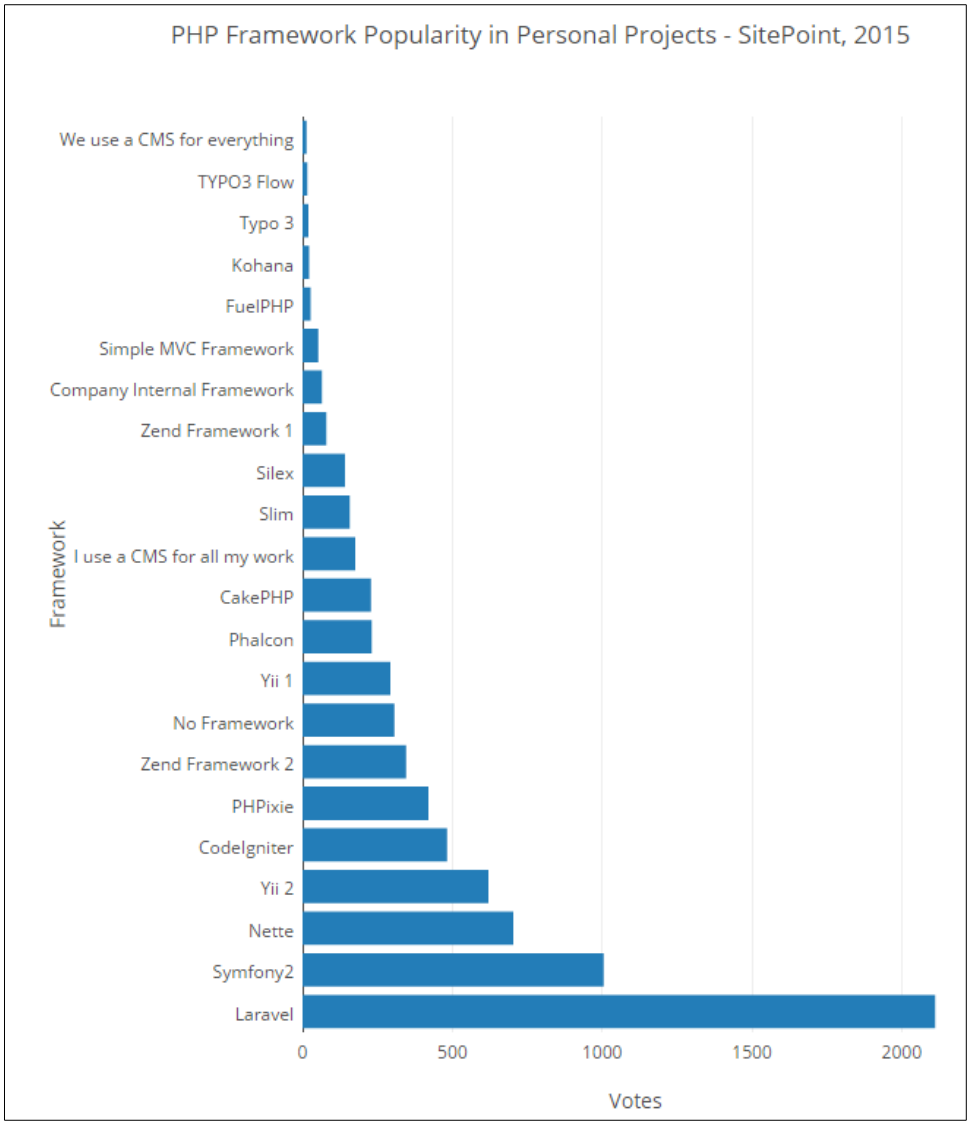
Yksinkertaistettuna ohjelmistokehys on joukko ennalta laadittuja koodipätkiä, jotka tarjoavat lyijykynällä tehdyn luonnoksen, jonka päälle valmis sovellus maalataan. Jossain tapauksessa kehys on yksinkertaisesti vain kokoelma erilaisia PHP-luokkia. Kehittäjällä on kuitenkin velvollisuus käyttää niitä oikein. Muissa tapauksissa saavutetaan intiimimpi suhde sovelluspalvelimen kanssa, mikä tarkoittaa, että niiden noudattaminen on pakollista, eikä vain suositeltavaa. Ohjelmistokehykset ovat lähes aina ilmaisia ja ne noudattavat usein myös avoimen lähdekoodin periaatteita. Kehysten ominaisuudet vaihtelevat laajalti, mutta ne kaikki yrittävät käsitellä joitakin ydintavoitteita. Kehyksien käyttäjillä voi olla paljon eri vaatimuksia ja jotkin kehykset toteuttavat enemmän ratkaisuja mitä toiset. (Lecky-Thompson & Nowicki 2009. 355–357)

Kehykset voivat poiketa toisistaan myös monella muulla tavalla. Esimerkiksi tiedostorakenteet tai vaikkapa koodin erottelu voivat poiketa täysin kehysten välillä. Arkkitehtuuriratkaisut ovat myös iso osa kehyksen identiteettiä, eli esimerkiksi vaikka suositun MVC arkkitehtuurin noudattamista kehyksessä.(Lecky-Thompson & Nowicki 2009. 355–357.). Tässä kappaleessa vertaillaan joukkoa sovelluskehyksiä ja tehdään vertailun perusteella kehyksen valinta.

5.1 Vertailtavat kehykset

Vertailtavaksi kehyksiksi valittiin nopean tutkimisen perusteella arvostetun ja tunnetun SitePoint.com-sivuston kyselyssä menestyneitä PHP-ohjelmistokehyksiä. Vertailtaviksi kehyksiksi valittiin Laravel, CodeIgniter, Symfony 2, CakePHP ja Phalcon. Kehyksiä yhdistää tietyt perustoiminnot ja toiminnallisuus, mutta samalla ne antavat myös vapautta ja muokattavuutta tarpeitten ja päämäärien tavoittamiseksi. Näiden kehysten vertailu osoittautui melko suoraviivaiseksi, koska niistä löytyi vertailua varten tehty yhteinen infografiikka. Infografiikan, eli havainnollistavan kuvallisen esityksen, oli taas tehnyt sivusto nimeltä Webhostface käyttäen hyötynä SitePointin kyselyssä saatuja tilastoja, Googlea ja Githubin trendejä. Infografiikasta poimittujen tietojen ja monien erilaisten Google hakujen perusteella saatiin melko hyvä kuva kustakin vertailuun otetusta ohjelmistokehyksestä. (Kitipova 2015.)

Tarkastellaan seuraavaksi hiukan SitePointin tekemän kyselyn tuloksia havainnollistavaa kuvaa PHP-kehysten suosiosta (KUVIO 2):



KUVIO 2: Php-kehysten suosio (mukaillen Skvorc 2015.)

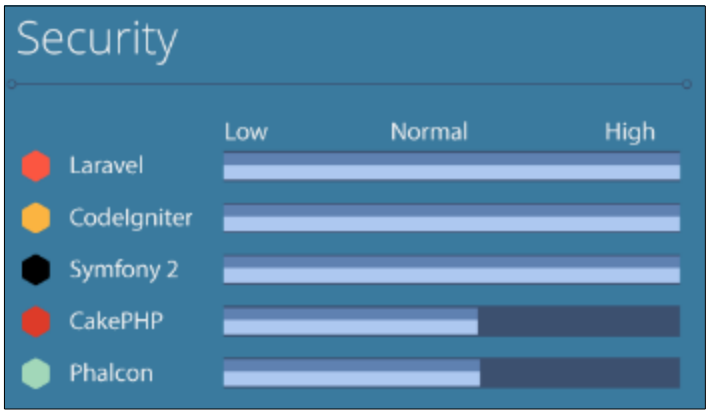
Kuvasta selviää heti kuinka suuren suosion Laravel on lyhyessä ajassa saavuttanut. Laravel sai kyselyssä yli 2000 ääntä ja toiseksi sijoittunut Symfony-kehys sai vain noin 1000 ääntä. Jos esimerkiksi Yii 1 ja Yii2 äänet yhdistettäisiin niin ei niist ä siltikään olisi haastamaan Laravelia sen suosiossa. Kyselyä ei tietenkään voida pitää absoluuttisena totuutena, mutta se on silti jonkin verran suuntaa antava ja siitä voi päätellä ainakin päällisin puolin suosiota. Osasyynä suosiota selittää varmasti myös uutuudenviehätys, koska Laravel on kuitenkin vielä

suhteellisen uusi ohjelmistokehys.

5.2 Vertailu

Suosion lisäksi tulee ottaa huomioon myös olennaisia asioita kuten tietoturva, nopeus, monipuolisuus ja käyttäjäkuntien määrä. Kaikenlainen käyttäjien kritiikki on myös yksi tärkeimmistä vertailukriteereistä, mutta sitä on vaikea tiivistää, koska se on jakautunut moneen eri paikkaan.

Tietoturva on hoidettu kaikissa vertailtavissa kehyksissä hyvin, mutta erityisen hyvin se on hoidettu Laravel-, CodeIgniter- ja Symfony 2-kehyksissä, kuten kuvio 3 osoittaa:

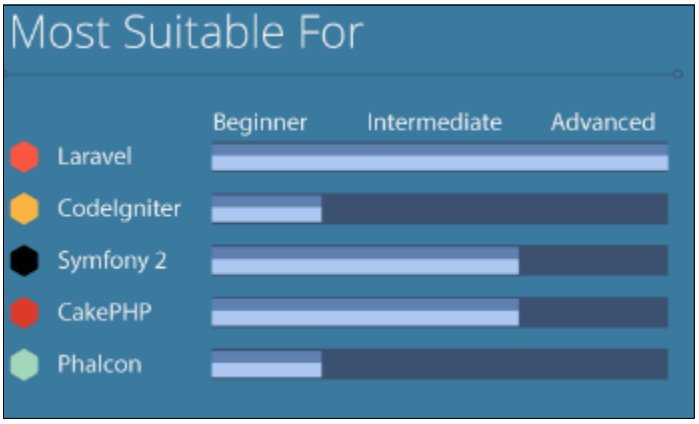


KUVIO 3: Vertailuun valittujen kehysten tietoturva (mukaillen Kitipova 2015)

Seuraavaksi tarkastellaan kehysten eroavaisuuksia nopeudessa. Kaikista vertailussa olleista kehyksistä nopeimmaksi kehykseksi selviää Phalcon. Seuraavaksi nopein on CodeIgniter ja siitä seuraavat loput kolme kehystä ovatkin keskenään yhtä nopeita. Phalconin arkkitehtuuriratkaisut tekevät siitä ehkä jopa markkinoiden nopeimman PHP-kehyksen. Phalconissa on muista vertailussa olevista kehyksistä poiketen käytetty PHP-ohjelmointikielen lisäksi myös C-ohjelmointikieltä. (Kitipova 2015.) Havainnollistetaan vielä sama seuraavalla kuviolla 4:

KUVIO 4: Vertailuun valittujen kehysten nopeuserot (mukaillen Kitipova 2015)

Muokattavuus ja monipuolisuus viittaavat yhdessä siihen, kuinka kokeneille kehittäjille mikäkin kehys on sopiva. Infograafin mukaan Laravel sopii ainoana vertailussa olevana kehyksenä kaikista kokeneimmille ja vaativimmille kehittäjille. Keskitason eli aloittelijoita enemmän osaaville kehittelijöille sopisi Laravelin lisäksi myös Symfony 2 ja CakePHP. Kaikista vähiten mahdollisuuksia vaativille kehittäjille vertailun kehyksistä tarjoavat CodeIgniter ja Phalcon. Otetaan vielä sama kuvana havainnollistamaan (KUVIO 5):



KUVIO 5: Vertailussa olleiden kehysten sopivuus eri tasoisille kehittäjille (mukaillen Kitipova 2015.)

5.3 Valinta

Kehyksen valinta syntyi loppujen lopuksi melko nopeasti ja vaivattomasti. Valittu kehys oli Laravel. Laravelin laaja tarjonta ja kaikki sen saama positiivinen palaute teki valinnasta suorastaan helpon. Se vastasi myös kaikkiin vaatimuksiin ja kriteereihin täydellisesti. Laravelin nopeasti saama suosio ja huomio eivät myöskään voineet olla vain sattumaa, joten Laravel PHP-ohjelmistokehys oli ehdottomasti hyvä, ellei jopa paras valinta. Hyvänä merkkinä mainittakoon myös, että kovin paljoa negatiivista palautetta tai kritiikkiä ei Laravelista löytynyt. Löytyy toki niitäkin ihmisiä, jotka kiistävät Laravelin paremmuuden, mutta kiistatonta on ainakin se, että Laravel on onnistunut ainakin markkinoinnissa ja brändäyksessä paremmin kuin mikään muu moderni ohjelmistokehys. Näkyvyys ja suosio johtavat laajaan käyttäjäkuntaan ja koska Laravelin käyttäjillä on suuri mahdollisuus vaikuttaa sen suuntaan ja tuleviin ominaisuuksiin niin tällä tavoin varmistuu Laravelin valoisa tulevaisuus PHP-kehysten joukossa. Seuraavassa pääkappaleessa käsitellään Laravelia ja sen ominaisuuksia laajemmin ja yksityiskohtaisemmin.

5.4 Muita PHP-ohjelmistokehyksiä

Uskon, että kaikilla suosituilla kehyksillä on puolensa eikä parasta kehystä välttämättä voi oikeasti edes valita. Muita vertailun ulkopuolelle jääneitä, mutta kuitenkin suosittuja mainitsemisen ja tutustumisen arvoisia kehyksiä on ainakin Kohana, Yii 1 ja 2, PHPixie, Zend 1 ja 2, Nette sekä FuelPHP. Kehyksillä voi olla hiukan eri kohderyhmiä tai käyttäjäkunta voi erottua muuten vain jollain tavalla. Esimerkkinä SitePointin kyselyssä alle 18-vuotiaat äänestivät eniten PHPixien puolesta, vaikka kaikissa muissa ikäryhmissä Laravel oli selvä voittaja eikä PHPixie kehys ollut lähelläkään kärkeä. Kaikilla suosituilla kehyksillä on varmasti omat hyvät sekä huonot puolensa, eli on tärkeää käyttää tilanteeseen sopivaa kehystä. Eli yksinkertaisemmat ja helppokäyttöisemmät kehykset ovat varmasti omaa luokkaansa esim. pieniä henkilökohtaisia projekteja varten, mutta vastaavasti suuriin projekteihin voidaan tarvita ominaisuuksia mitä vaatimattomammat kehykset eivät välttämättä tarjoa.

6 LARAVEL

Laravelin suosio on saanut räjähdysmäisen suosion matkallaan yhdeksi suosituimmista ja laajimmin käytetyistä PHP-kehyksistä hyvin pienessä ajassa. (Bean 2015. 1.) Laravelin kehittäjä Taylor Otwellin filosofia on painottanut kahta arvoa Laravelin kehityksessä ja ne ovat ohjelmistokehittäjän nopeuden ja onnellisuuden lisääminen. Hänen mukaansa iloinen ohjelmistokehittäjä kirjoittaa parasta koodia. On selvää, että kaikki työkalut tai kehykset välittävät kehittäjien iloisuudesta tai onnellisuudesta, mutta Laravelin kehittämisessä se on ollut nimenomaan etusijainen päämäärä, eikä toissijainen niin kuin kehyksillä yleensä. Toiset kehykset voivat kohdistaa arkkitehtuurisen puhtauden ensisijaiseksi päämääräksi tai, mutta Laravel on keskittynyt pääasiassa yksilöllisen kehittäjän palvelemiseen. Laravel auttaa ideoiden toteuttamisessa ilman koodin tuhlausta käyttäen moderneja standardeja ja tehokkaita työkaluja. (Stauffer 2016. 10–14.)

6.1 Historia

Laravelin tarina alkoi vuonna 2011. Laravelin kehittäjän Taylor Otwellin mukaan Laravelin ensimmäinen versio eli versio 1 beta julkaistiin kesällä 2011 yksinkertaisesti vain hänen aikaisemmin käyttämänsä CodeIgniter PHP-kehyksen kasvukipujen ratkaisemiseksi. Vaikka CodeIgniter oli pitkän aikaa vuosien 2009–2011 yksi suosituimmista ellei jopa suosituin PHP-ohjelmistokehys, oli siinäkin omat puutteensa. Otwell koki, että CodeIgniterista puuttui paljon websovelluksien rakentamiseen vaatimia toiminnallisuuksia kuten esim. sisäänrakennetut kirjautumis- ja rekisteröitymisominaisuudet. (Surguy 2013.)

6.1.1 Laravel 1

Jo heti ensimmäisen Laravel version toiminnallisuus oli melko vakuuttavaa. Laravel esitteli ominaisuuksinaan sisäänrakennetun kirjautumisen, Eloquent ORM-toiminnallisuuden tietokantaoperaatioille, lokalisoinnin, mallit sekä riippuvuudet, yksinkertaisen reititysmekanismin, välimuistin, sessiot, näkymät, laajennettavuuden moduulien ja kirjastojen kautta, lomake ja HTML aputoiminnot sekä joitain muita ominaisuuksia. Tässä vaiheessa Laravel ei vielä ollut

MVC-arkkitehtuuri, koska se ei tukenut vielä kontrollereiden käyttöä. Siitä huolimatta kehittäjät rakastuivat välittömästi Laraveliin sen puhtaan syntaksin ja uuden kehyksen potentiaalin takia. Tulevien kuukausien aikana Taylor lisäsi kehyksen toiminnallisuutta, kuten validaatiometodeja, sivunumerointiominaisuuden ja komentorivipakettiasentajan. Alle kuudessa kuukaudessa ensimmäisen julkaisun jälkeen julkaistiin Laravelin toinen versio. (Surguy 2013.)

6.1.2 Laravel 2

Laravelin kakkosversio julkaistiin yksityisesti kehittäjille 24.11.2011. Julkaisun mukana tulleessa ”lue minut”-tiedostossa todetaan näin: ”Vapauttaen sinut spagettikoodista, Laravel auttaa sinua rakentamaan loistavia sovelluksia käyttäen yksinkertaista, ilmeikästä syntaksia. Kehittämistyön pitäisi olla luova kokemus josta nautit, eikä jotain joka on tuskallista. Nauti raittiista ilmasta.”. Laravelin toisen merkittävän päivityksen mukana tuli muutama vahva ominaisuus kuten kaivattu tuki kontrollereille ja Laravelin oma ”Blade”-sivumoottori. Tuki kontrollereille teki Laravelista myös täysin pätevän MVC-kehyksen. Kehittäjät olivat tyytyväisiä siihen, minkälaiseksi Laravel oli muovautumassa, mutta vielä kuitenkin tyytymättömiä kolmannen osapuolen moduulien tukemiseen. Laravelin jo tuolloin loistavan toiminnallisuuden ansiosta se houkutteli paljon uusia kehittäjiä kokeilemaan Laravelia. Tästä vain alle kaksi kuukautta myöhemmin julkaistiin Laravelin kolmas suuri päivitys, eli Laravel 3. (Surguy 2013.)

6.1.3 Laravel 3

Laravelin kolmas versio julkaistiin helmikuun lopulla. Samalla Laravelin omat kotisivut päivitettiin ja Laravelin uusina ominaisuuksina esiteltiin muun muassa ”artisan”-niminen komentorivin rajapinta, tietokantojen migraatiot, tapahtumaominaisuudet, lisää sessio- tietokantaohjaimia, ulkopuolisten moduulien integroiminen, eli ”bundle”-ominaisuus. Myös aiemmin lisätty ominaisuus Eloquent luettiin väliaikaisesti ulkopuoliseksi moduuliksi eli bundle-osaksi, eli sitä ei sisällytetty Laravel-projektin oletusasennukseen. Myöhemmissä Laravelin kolmosversion päivityksissä se kuitenkin luettiin takaisin Laravelin ominaisuuksiin ja se on pysynyt sellaisenaan tähän päivään asti. Laravel 3 oli kaikista vakain ja toimivin Laravel-julkaisu joka oli tar-

peeksi tehokas käytettäväksi moniin erilaisiin websovelluksiin. Se tarjosi yksinkertaisuutta ja erittäin lyhyttä oppimiskäyrää verrattuna muihin sen aikaisiin PHP-ohjelmistokehyksiin. Laravel alkoi saavuttaa sen ajan suurimpia kehyksiä kuten esim. CodeIgniteria ja Kohanaa. Monet kehittäjät alkoivat vaihtaa muista kehyksistä Laraveliin sen tehokkuuden ja ilmaisevuuden takia. Noin 5 kuukautta Laravelin kolmannen suurjulkaisun jälkeen sen kehittäjä alkoi kirjoittaa uudelleen koko kehystä alusta alkaen. Sen seurauksena syntyi koodinimi Illuminate, eli Laravel 4. (Surguy 2013.)

6.1.3 Laravel 4 (Illuminate)

Laravel sai uusia versioita aina parin kuukauden välein. Laravel 4 julkaistiin virallisesti yhden vuoden ja kolmen kuukauden sen kolmannen version julkaisemisesta eli toukokuun 28. päivä vuonna 2013. Vaikka päivitysten tiheys viittaa yleisesti siihen, että kehys kehittyy, niin päivitysten määrä söi samalla Laravelin uskottavuutta. Laravelia kommentoitiin liian nopeatempoiseksi ja epävakaaksi, koska kehittäjät joutuivat tekemään migraatioita uusiin versioihin ja isompien sovellusten kohdalla se ei aina ollut mahdollista. Siirtymä aikaisemmin käytetystä arkkitehtuurista uudempaan ei aina ole helppoa tai edes mahdollista. Koska ne oli rakennettu aikaisemmin käytettyjen arkkitehtuurien päälle. Laravelin käyttäjäkanta pyysi vakauden lisäämistä, joitain uusia ominaisuuksia ja parempaa yksikkötestausta Laravelin komponenteille. (Surguy 2013.)

Laravel 4 oli kaiken aiemman kehitystyön huipentuma ja osa PHP-kehityksen valoisaa tulevaisuutta. Laravel 4 kirjoitettiin alusta asti uusiksi eri komponenttien kokonaisuutena jotka integroituvat keskenään luodakseen kehyksen. Näiden komponenttien hallinta hoidetaan parhaan saatavilla olevan PHP riippuvuuksienhallintatyökalulla nimeltään ”Composer”. Laravelin neljäs versio tarjoaa laajan joukon ominaisuuksia, joita mikään muu Laravelin versio tai jopa mikään muu PHP-kehys ei tarjoa. Aikaisemmasta poikkeavana Laravelin uudemmat julkaisut noudattavat säännöllisempää julkaisuaikataulua, eli päivityksiä puolen vuoden välein aikaisemman epäsäännöllisyyden sijaan. (Surguy 2013.)

6.1.4 Laravel 5

Laravelin 4.3 versio oltiin suunniteltu julkaistavaksi marraskuussa 2014, mutta kehittämisen edetessä kävi kuitenkin selväksi, että muutosten merkittävyys ansaitsi merkittävämmän julkaisun. Siitä johtuen Laravel 4.3 sijaan julkaistiinkin Laravel 5 helmikuussa 2015. Laravel toi mukanaan muun muassa hiotun hakemistorakenteen, sopimusrajapintojen esittelyn, Socialite-toiminnallisuuden sosiaalisen median kautta kirjautumiseen, Elixir-ominaisuuden ja uuden komentorivikäyttöliittymän. (Stauffer 2016.)

6.1.5 Julkaisuaikataulu

Vuonna 2013, Laravelin kehittäjä Taylor Otwell julkaisi ensimmäisen virallisen julkaisusyklin kehykselleen. Aikaisemmin julkaisuja tuli epäsäännöllisesti jopa parin kuukauden välein, mutta uudessa julkaisumallissa Laravel päivittyy 2 kertaa vuodessa, eli puolen vuoden välein. Toinen päivityksistä julkaistaisiin aina kesäkuun tienoilla ja toinen taas aina loppuvuodesta. Tämän työn kirjoittamishetkellä Laravelin uusin versio oli 5.2 versio. Laravelin kehittäjän julkaiseman, eli jo aikaisemmin mainitun aikataulun mukaisesti sen 5.3 versio on suunniteltu julkaistavaksi kesäkuussa 2016. Sitä seuraava, eli versio 5.4 on suunniteltu julkaistavaksi joulukuussa 2016. Työn kirjoittamishetkellä viimeinen aikatauluun merkitty versio on Laravel 5.5 ja sen oletetaan tulevan julkaistuksi kesäkuussa 2017. (Barnes 2016.)

6.2 Ominaisuudet

Ohjelmistokehittäjällä on usein paljon laajasti tunnettuja ongelmia, joihin kehys tarjoaa ratkaisujaan, eli suurin osa näistä ominaisuuksista siis pyrkii löytämään parhaan tavan ratkaista jokin ongelma. Kokonaisuudessaan Laravelin ominaisuuksia on pitkä lista, joten niitä kaikkea olisi vaikea esitellä kerralla. Tässä osiossa esitellään ja tarkastellaan valitsemiani Laravelin ominaisuuksia. Kaikkia mainittuja ominaisuuksia testattiin myös käytännössä tätä opinnäytetyötä tehdessä.

6.2.1 Reititys

Aloittelevilla PHP-kehittäjillä ei usein ole tuntemusta kuin aivan perusreititystavoista. Onneksi Laravelin tapa hoitaa reitittäminen on yksinkertainen sekä samalla helppokäyttöinen, eli uuden tavan opettelu ei ole kovinkaan vaikeaa. (Way 2012.)

Esimerkki havainnollistaa, kuinka helposti käyttäjätiedot voidaan esittää tehokkaasti ja joustavasti Laravelin reititysominaisuuden avulla. Wayn esimerkissä käyttäjän pyyntö osoitteeseen ”esimerkkisivu.com/users/1” antaa hänelle näkymän ”profile.php”, joka sijaitsee ”users” hakemistossa ja näyttää hänelle juuri oikeat profiilitiedot reititysominaisuuden avulla. (KUVIO 6)

```
1  Route::get('users/{id}', function($id) {
2      // find the user
3      $user = User::find($id);
4
5      // display view, and pass user object
6      return View::make('users.profile')
7          ->with('user', $user);
8  });
```

KUVIO 6: Laravelin reititysominaisuus (mukaillen Way 2012.)

Projektin laajentuessa reititystiedosto voi helposti alkaa täyttyä liialla määrällä koodia, eli siitä voi tulla hyvinkin vaikeasti luettava. Kun sivusto saa tarpeeksi tällaista toiminnallisuutta niin kannattaa alkaa harkitsemaan kontrollereiden käyttöä. Kontrollerit sijaitsevat hakemistossa ”Controllers” ja niitä voi lisätä sinne aina tarvittaessa. Aiemman esimerkin saman logiikan voi siis myös sisällyttää kontrollereihin, eli tässä tapauksessa reititystiedostossa viitataan vain kontrolleriin nimeltä ”users” ja pyydetään käyttämään ”show” metodia. Näin näkymän esittäminen on käyttäjäkontrollerin vastuulla. (Way 2012.) Seuraavassa Wayn esimerkissä kuvan yläosa on reititintiedoston osa ja kuvan alaosa on käyttäjäkontrollerin rakenne (KUVIO 7):

```
1  Route::get('users/{id}', 'Users@show');
```

```
<?php

class UsersController extends Controller {
    /**
     * Display the specified resource.
     */
    public function show($id)
    {
        // find the user
        $user = User::find($id);

        // display view, and pass user object
        return View::make('users.profile')
            ->with('user', $user);
    }
}
```

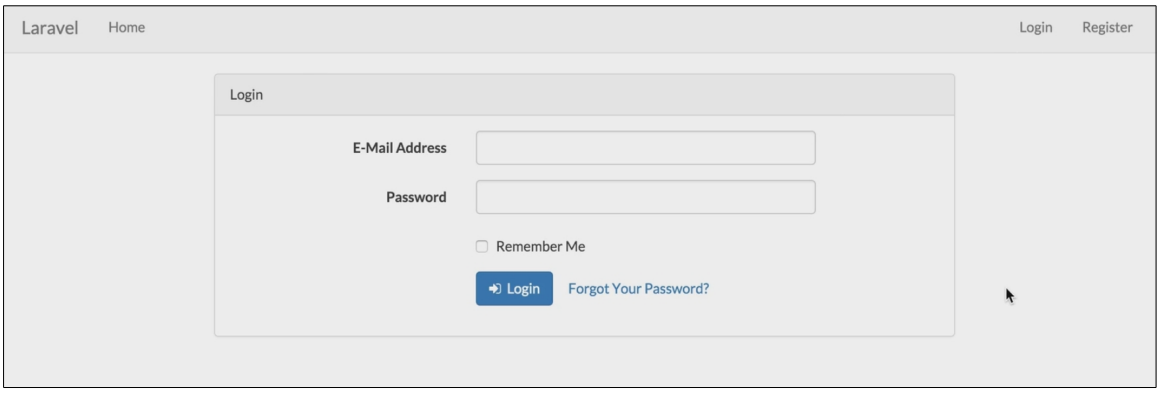
KUVIO 7: Laravelin reititysominaisuus kontrollereita käyttäen (mukaillen Way 2012.)

6.2.2 Kirjautuminen, rekisteröityminen ja salasanan palautus

Yksi Laravelin käytetyimmistä ominaisuuksista on sen sisäänrakennettu kirjautumisominaisuus. Kirjautumisominaisuuden rakentaminen jokaiselle projektille erikseen on melko työlästä ja aikaa vievää puuhaa ja kokemattomammalle sovelluskehittäjälle jopa yllättävänkin haasteellista, koska siihen liittyy samalla myös paljon tietoturvaankin liittyviä riskejä. Laravelin tarjoamalle turvalliselle ja muutenkin hyvin rakennetulle pohjalle onkin siis varmasti käyttöä sekä pieniessä, että suuremmissakin sivustoprojekteissa. Laravelin kirjautumisominaisuus sisältää myös rekisteröinnin ja salasanan palautuksen sekä erilliset tarkistukset ja virheilmoitukset, jos esim. kirjautumistiedot eivät täsmää. Vaikka kieli on oletuksena englanti, niin eri kielen lisääminen on suhteellisen helppoa. Kirjautumisominaisuus on suhteellisen helppokäyttöinen ja yllättävän nopeasti käyttöön otettava. Laracast.com-sivuston opetusvideossa Jeffrey Way opastaa kirjautumisominaisuuden käyttöä videosarjan ”Laravel 5 from scratch 13. jaksossa nimeltä ”Authenticate Your Users”.

Laravelin sisäänrakennetun kirjautumisominaisuuden saa käyttöön komentorivissä Laravelin artisan komennolla ”php artisan make:auth”. Komento generoi yhteensä 8 eri näkymää eri tarkoituksiin kuten esim. rekisteröintiä tai salasanan palautusta varten. Sama komento myös luo uuden kontrollerin nimeltään HomeController ja päivittää reititystiedostoa hieman. Way suositteleekin videossaan ”make:auth” komennon käyttöä heti uuden projektin alkuun. Komennon lisäksi pitää vielä määrittää käytettävä tietokanta ja yksinkertaisuuden takia hän käyttää esimerkissään sqlite-tietokantaa. Tietokanta määritetään tietokannan konfiguraatiotiedostossa ”database.php”, joka löytyy hakemistosta config. Konfiguraatiotiedostossa pitää määrittää vain oletustietokanta ja tarvittaessa myös tietokannan yksityiskohtaiset tiedot. Wayn opetusvideon esimerkissä riitti kuitenkin vain oletusarvon muuttaminen mysql-tietokannasta sqlite-tietokantaan ja tyhjän sqlite-tietokantatiedoston luominen. Tietokannan määrittämisen jälkeen piti vielä suorittaa Laravelin migrate-ominaisuus komennolla ”php artisan migrate” sekä viimeiseksi ”php artisan serve”-komento. Ympäristötiedoston ”.env” ympäristömuuttujia voi muokata kehitysympäristön tarpeen mukaan ja videon esimerkissä muutettiinkin MAIL_DRIVER kohtaan oletusasetus smtp:n sijaan log, eli laravelin lokitiedoston käyttöä. Sen lisäksi config hakemiston ”mail.php”-tiedostoa muutettiin vielä hiukan salasanan palautuksen käyttöä ja testaamista varten, mutta tämän vaiheen voi käytännössä jättää tekemättäkin. Näiden vaiheiden jälkeen kirjautuminen, rekisteröityminen ja salasanan palautus saatiin helposti ja nopeasti käyttövalmiiksi. (Laravel from scratch, episodi 13: Authenticate Your Users. Laracast 2016)

Oletusulkoasu ja toiminnallisuus sopii hyvin projektin testaamiseen, mutta kaikkea voi tarvittaessa mukauttaa projektin vaatimusten mukaan. Laravelin oletusulkoasu käyttää apunaan Twitterin kehittämää suosittua bootstrap css-kehystä, mutta se ei ole millään tavalla pakote tai vaatimus. Seuraavassa kuvassa on täysin muokkaamaton käyttövalmis oletuskirjautumisnäkymä (KUVIO 8):



KUVIO 8: Laravelin kirjautumisominaisuuden oletusnäkymä

6.2.3 Blade sivumoottori (Templating engine)

PHP-kielessä on omat ongelmansa, ja siitä syystä suurin osa nykyaikaisista ohjelmistokehyksistä tarjoavat jonkin sivumoottorin niiden ratkomiseen. Useimmissa Symfony-pohjaisissa kehyksissä käytetään suosittua Twig-nimistä sivumoottoria, mutta tässä Laravel tekee poikkeuksen vaikka Symfony-kehys on senkin pohjana. Laravel mahdollistaa, mutta ei kuitenkaan pakota käyttämään yhtä sen tehokkainta ominaisuutta eli blade sivumoottoria. Blade sai inspiraation alunperin asp.NET Razor moottorista. Se muistuttaa kovasti Twig-moottoria, mutta sen syntaksi on lähempänä Razor-moottoria.

Sen lisäksi Laravelin Blade-moottorin oppimiskäyrä on matalampi kuin Twig-moottorin opettelussa. Haluttaessa on myös mahdollista käyttää Twigiä tai jotain muuta sivumoottoria Laravelin Bladen sijaan.(Stauffer 2016. 53)

Stauffer havainnollistaa, miltä sama toiminnallisuus näyttää eri sivumoottoreilla koodattuna. Esimerkissä tulostetaan kaikkien käyttäjien etu ja sukunimet ja jos käyttäjiä ei ole, tulostetaan teksti ”No users.”. Esimerkissä ylin syntaksi on PHP:lla toteutettu, keskimmäisenä Symfony-kehyksissä suosittu Twig ja alimpana Laravelin Blade-moottori (KUVIO 9):

```
<?php /* PHP */ ?>
<?php if (empty($users)): ?>
    No users.
<?php else: ?>
    <?php foreach ($users as $user): ?>
        • <?= $user->first_name ?> <?= $user->last_name ?><br>
    <?php endforeach; ?>
<?php endif; ?>
```

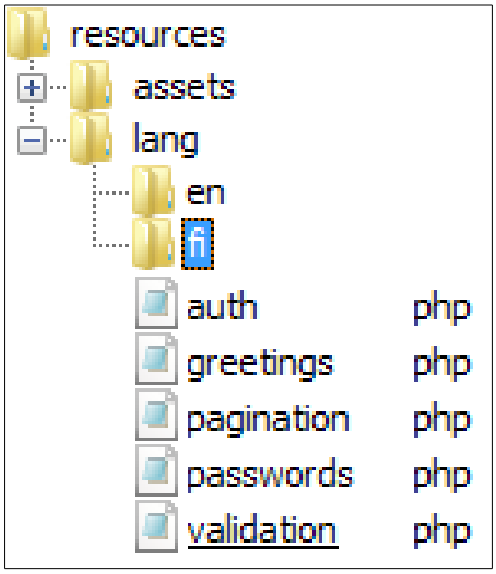
```
{# Twig #}
{% for user in users %}
    • {{ user.first_name }} {{ user.last_name }}<br>
{% else %}
    No users.
{% endfor %}
```

```
{{-- Blade --}}
@forelse ($users as $user)
    • {{ $user->first_name }} {{ $user->last_name }}<br>
@empty
    No users.
@endforelse
```

KUVIO 9: Syntaksien vertailu: PHP - Twig - Blade. (mukaillen Stauffer 2016.)

6.2.4 Lokalisointi

Laravelin ratkaisu monikieliseen sivustoon oli hyvin toteutettu ja helposti lähestyttävä: Laravelin lokalisoinnin (Engl. localization) avulla saa määritettyä vaikka eri kielille räätälöidyt virheilmoitukset tai tervehdysviestit erikielisille käyttäjille. Kaikki on muokattavissa ja kun tähän ominaisuuteen tutustui syvemmin, niin myös jopa eri kielien monimutkaiset kielioppisäännöt oli mahdollista ottaa huomioon erityistilanteissa, kuten vaikka räätälöityjä virheilmoituksia laatiessa. Laravelin lokalisoinnissa käytetään valmiiksi generoituja kielitiedostoja, jotka määrittävät miten sivusto toimii ja mitä viestejä käyttäjä saa milläkin kielellä. Uusille kielille voi tehdä uusia kielikansioita tarvittaessa, eli sivusto voi käytännössä olla käännetty vaikka kymmenille eri kielille. Kuvassa esitetään kielikansioiden tiedostorakenne (KUVIO 10).



KUVIO 10: Laravel lokalisoinnin tiedostorakenne ja oletustiedostot (mukaillen Way 2012.)

6.2.5 Sähköpostin lähettäminen

Sähköpostien lähettäminen sovelluksen kautta on aina ollut hankalampaa mitä voisi ensiksi olettaa. Laravel ratkaisee tämän yleisen ongelman käyttämällä suosittua ”SwiftMailer”-pakettia kehysratkaisunsa pohjana.

Aluksi määritetään sähköpostin palvelutarjoajan arvot ”config/mail.php”-tiedostoon joka näyttää tältä (KUVIO 11):

```
<?php
// app/config/mail.php

return array(
    'host' => 'smtp.example.com',
    'port' => 2525,
    'from' => array('address' => null, 'name' => null),
    'encryption' => 'tls',
    'username' => null,
    'password' => null,
);
```

KUVIO 11: Sähköpostin lähetyksen konfigurointi. (mukaillen Way 2012.)

Seuraavaksi tehdään näkymä, joka halutaan näyttää, kun uusi jäsen rekisteröityy sivustolle. Wayn esimerkissä luodaan yksinkertainen näkymä (KUVIO 12):

```
<?php
// app/views/emails/welcome.blade.php

<html>
    <body>
        Hi there, {{ $user->name }}. Thanks again for signing up for t

        Thanks,
        Management
    </body>
</html>
```

KUVIO 12: Luotu näkymä. (mukaillen Way 2012.)

Kun konfigurointi ja haluttu näkymä on hoidettu, tehdään vielä varsinainen reititys, jossa lähetetään sähköposti rekisteröityyn sähköpostiin. Sähköpostiin saa helposti ja vaivattomasti viestin ja otsikon ja tarvittaessa vaikka liitetiedoston. Reitittäminen voisi näyttää esim. tältä (KUVIO 13):

```
Route::get('/', function()
{
    $user = User::find(1);
    $data = [ 'user' => $user ];

    // email view, data for view, closure to send email
    Mail::send('emails/welcome', $data, function($message) use($user)
    {
        $message
            ->to($user->email)
            ->subject('Welcome Bieber Fan!')
            ->attach('images/bieberPhoto.jpg');
    });

    return 'Welcome email sent!';
});
```

KUVIO 13: Sähköpostiviestin reitittäminen (mukaillen Way 2012.)

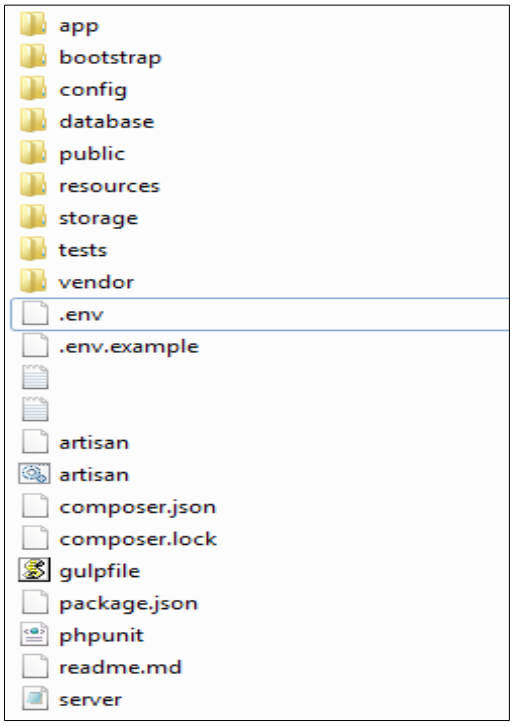
SwiftMailerin ja Laravelin tarjoama toiminnallisuus on omaa luokkaansa ja tekee monimutkaisistakin toiminnoista paljon helpompia toteuttaa. Niiden ansiosta sähköpostin lähettäminen on Laravel-kehys ympäristössä juuri niin helppoa kuin sen pitäisikin olla.

6.2.6 Muut

Laravelissa on siis myös paljon muita ominaisuuksia joita ei käyty läpi. Näitä mainitsemisen arvoisia tehokkaita ominaisuuksia on ainakin Laravelin tietokantojen migraatiot, Eloquent ORM (Engl. Object-relational mapping), Composer paketinhallintatyökalu ja ”Tinker”-työkalu. Myös kaikkiin näihin ominaisuuksiin törmättiin tätä opinnäytetyötä tehtäessä. Näidenkin lisäksi on paljon sellaisia hyödyllisiä ominaisuuksia joihin ei kuitenkaan tämän opinnäytetyön edetessä satuttu ainakaan tietoisesti käyttämään. On myös paljon ominaisuuksia jotka eivät ilmoita läsnäolostaan millään tavalla, vaan kaikki tapahtuu näkymättömissä, eikä niistä tarvitse välttämättä sen enempää huolehtiakaan. Kunhan vain tietää, että Laravel tekee paljon hyödyllisiä asioita puolestasi.

6.4 Tiedostorakenne

Tässä osuudessa käydään läpi Laravelin tiedostorakennetta. Kaikilla tiedostoilla on tietenkin jokin tarkoitus, mutta suurin osa Laravelin tiedostoista toimii tai on käyttövalmis sellaisenaan heti asennuksen jälkeen eikä niitä tarvitse muokata muuta kuin erityistarpeen mukaan. Laravelin aiemmissa kuten esim. Laravel 4:n tiedostorakenne oli hiukan erilainen verrattuna uudemman Laravel 5:n hiukan parannettuun tiedostorakenteeseen. Juuri asennetun Laravel 5-projektin tiedostorakenne on seuraavan kuvion 14 mukainen:



KUVIO 14: Laravel-projektin tiedostorakenne asennuksen jälkeen

6.4.1 Irtotiedostot

Juuri asennettuun Laravel projektiin kuuluu paljon tiedostoja jotka eivät ole missään kansiossa vaan heti Laravel-projektin tiedostojuuressa. Näissä tiedostoissa on suurimmaksi osaksi konfiguraatiotiedostoja eri lisäosille tai käytetyille ominaisuuksille, mutta myös joitain poikkeuksia on.

Ympäristömuuttujat tallennetaan ”.env”-tiedostoon. Esimerkkinä ”.env”-tiedoston käytöstä eri ympäristöille myös ”.env.example” on sisällytetty vasta asennettuun Laravelin projektiin. Git versionhallintajärjestelmä jättää nämä tiedostot huomiotta, koska voidaan olettaa, että ympäristöillä on aina eri ympäristömuuttujat ympäristöstä riippuen. Tiedostot ”.gitignore” ja ”.gitattributes” ovat konfiguraatiotiedostoja Git versionhallintajärjestelmää varten. Tiedosto nimeltä ”artisan” mahdollistaa taas artisan komentojen käytön komentorivillä. (Stauffer 2016.)

”Composer.json” ja ”composer.lock” ovat Composer-riippuvuushallintatyökalun konfiguraatio-

tiedostoja. ”Composer.json” on vapaasti käyttäjän muokattavissa, kun taas ”composer.lock” on käyttäjältä täysin lukittu. Projektissa käytetyt sovellusriippuvuudet löytyvät näistä tiedostoista ja niitä voi lisätä projektin kasvaessa. Myös ”package.json” on vastaavanlainen konfiguraatiotiedosto, mutta vain frontend asseteille. ”Gulpfile.js” on valinnainen konfiguraatiotiedosto Laravelin omalle Elixir-ominaisuudelle ja gulp-tehtävänajaja työkalulle. ”Phpspec.yml” ja ”phpunit.xml” ovat konfiguraatiotiedostoja erillisille testaustyökaluille. ”Readme.md” sisältää lyhyen esittelyn Laravel-ohjelmistokehykseen. ”Server.php” on varapalvelintiedosto, joka on tehty vaihtoehdoksi vähemmän ominaisuuksia tukeville servereille, jotta ne pystyvät silti näyttämään edes jonkinlaisen Laravel sovelluksen näkymän. (Stauffer 2016.)

6.4.2 Hakemistot

Suurin osa itse sovelluksesta sijoitetaan ”app”-hakemistoon. Eli sinne varastoidaan kaikki kontrollerien, mallien, reittimäärittelyjen, komentojen ja PHP-verkkotunnusten koodi. (Stauffer 2016.)

Hakemistossa nimeltä ”bootstrap” pidetään Laravel-kehyksen omien bootstrap-toiminnon ja automaattisen latauksen konfiguraatiotiedostoja. Siellä säilytetään myös muutamaa Laravelin generoimaa tiedostoa Laravelin bootstrap-ominaisuuden suorituskyvyn optimointiin. Lyhyesti sanalla ”bootstrap” tarkoitetaan asioiden rekisteröimistä Laravelissa. (Otwell 2016a.)

Selvennettäköön siis vielä, että ”Bootstrap” sanalla ei siis tässä yhteydessä viitata Twitterin kehittämään samannimiseen CSS-sovelluskehykseen, vaan Laravelin omaan ominaisuuteen.

Loput Laravelin käyttämistä konfiguraatiotiedostoista sijoitetaan hakemistoon nimeltä ”config”. Kaikki tietokantojen tiedostot säilytetään oletuksena ”database”-hakemistossa. ”Public”-hakemistossa on kaikki tiedostot, joihin serveri viittaa kun se esittelee sivuston. Siellä säilytetään myös kaikki kuvat, tyylitiedostot, skriptit ja ladatattavat tiedostot, eli siis kaikki julkisesti esillä olevat tiedostot. Kaikki tiedostot joita skriptit tarvivat toimiakseen sijoitetaan ”resources”-hakemistoon. ”Resources”-hakemisto sisältää myös ”views”-hakemiston jossa kaikki sovelluksen näkymät ovat. Myös kaikki kieliasetuksiin liittyvät tiedostot säilytetään ”resources”-

hakemistossa. Hakemukseen ”storage” varastoidaan paljon erinlaisia kehyksen generoimia tiedostoja kuten esim. lokitiedostot tai mallinnetut blade-tiedostot. Kaikki automatisoidut testit sijoittuvat ”tests”-hakemistoon. Kaikki Composer riippuvuustyökalun riippuvuudet sijaitsevat ”vendor”-hakemistossa, mutta samoin kuin ”.env” tiedostojen kanssa git sivuuttaa myös koko ”vendor”-hakemiston versiohallinnallisista syistä. (Stauffer 2016.)

6.5 Laracasts.com

Laracasts.com on suurin Laravel opetusvideoita tarjoava sivusto. Suurin osa videoista käsittelee siis Laravelia, mutta sivustolla on myös tarjolla paljon videoita jotka liittyvät vain etäisesti Laraveliin ja sen ominaisuuksiin. Laracasts.com-sivusto on maksullinen, mutta myös ilmaisia videoita muun muassa perusteista on laajasti tarjolla. Opetusvideoiden seasta löytyy siis myös paljon yleiseen ohjelmointiin liittyviä videoita ja tilaa on myös suosittujen aputyökalujen tai lisäosien käyttöön liittyville videoille. Web-ohjelmointiin liittyviä uusia aputyökaluja ja tekniikoita kehitetään koko ajan ja myös niiden käyttöä helpottavia uusia opetusvideoita ilmestyy Laracasts-sivustolle kiitettävää tahtia. Tämän opinnäytetyön valmistuessa opetusvideoita oli kaiken kaikkiaan jo yli 650 ja kaikki videot ovat todella laadukkaita, vaikka videot ovat vain yhden erityisen aktiivisen sovelluskehittäjän käsialaa. Laracasts.com-sivuston perustaja Jeffrey Way on siis yksi Laravelin isoimmista uranuurtajista, kun miettii, miksi Laravel on yksi menestyneimmistä nykyaikaisista ohjelmistokehyksistä. Sivuston opetusvideot eivät myöskään junnaa paikallaan vaan uusia opetusvideoita tulee viikoittain ja parhaimmillaan jopa parin päivän välein.

Vaikka Laracasts.com ei siis käytännössä ole Laravelin ominaisuus tai oppimisen vaatimus niin on se silti ehdottomasti mainitsemisen arvoinen ja kiistattomasti yksi Laravelin suurimmista vahvuuksista ja yksi sen menestymisen salaisuuksista.

7 CASE: VERKKOKAUPPA

Yksi työn päämääristä oli päästä kokeilemaan opittuja asioita ja ominaisuuksia Laravelista verkkosivujen suunnittelu ja toteuttamisprosessissa. Verkkosivujen aiheeksi tuli verkkokauppa, koska se on tarpeeksi monipuolinen Laravelin ominaisuuksien laajaan kokeiluun. Verkkokauppa on kuitenkin kokonaisuudessaan hyvinkin laaja ja monimutkainen toteuttaa, joten sen toiminnallisuutta piti rajoittaa työn edistyessä. Tärkein haluttu suurempi ominaisuus oli kirjautuminen ja ostoskorin toteutus, mutta myös pienempiä ominaisuuksia tuli käytettyä. Työn päämääränä oli lähinnä oppia uutta, eli oli myös suunnitelmana toteuttaa muitakin haluttuja tekniikoita, eikä ainoastaan Laravel-kehyksen ominaisuuksia. Vaikka verkkokauppaa jouduttiin rajoittamaan työn laajuuden vuoksi, niin tarkoituksena on kuitenkin jatkaa työtä tulevaisuudessa opinnäytetyön jälkeenkin ja saada aikanaan täydellisen valmis verkkokauppapohja.

7.1 Vaatimukset

Ohjelmistoprojektia aloittaessa kannattaa asettaa lopulliselle tuotteelle jonkinlaisia vaatimuksia, vaikka sivusto tulisikin vain testikäyttöön. Verkkokaupalla voi tilanteen mukaan olla hyvinkin yksilöllisiä tarpeita, eli esimerkiksi jotain erikoistuotetta myydessä voidaan vaatia sellaista toiminnallisuutta mitä verkkokaupalla ei yleisesti ole. Seuraavissa osioissa käydään läpi verkkokauppasivuston yleisistä vaatimuksista jakaen ne kahteen ryhmään eli toiminnallisiin ja ei-toiminnallisiin vaatimuksiin.

7.1.1 Toiminnalliset vaatimukset

Verkkokaupan yleisiä toiminnallisia vaatimuksia ovat ainakin rekisteröityminen, kirjautuminen, salasanan palautus, tuotesivut (mahdollisesti myös tuotteiden etsinnän helpottamiseksi etsintä tai rajausvalikoita), yksittäisen tuotteen tilaussivu tai ostoskori ja maksamisprosessi jossa voi olla monta vaihetta. Jos verkkokauppa tehdään asiakkaalle niin myös erilliset hallintasivut olisi hyvä toteuttaa. Hallintasivuille voi lisätä monenlaisia ominaisuuksia ja toimintoja asiakkaan resurssien ja tarpeiden mukaan. Tällaisia ominaisuuksia voi olla esimerkiksi tuotteiden

lisääminen sivuston käyttöliittymän kautta tai vaikka uutisten ja muiden sivujen päivittämisen mahdollistaminen käyttöliittymän kautta. Tilastojen näyttäminen pääkäyttäjälle olisi myös hyvä esimerkki tällaisesta hallintasivun ominaisuudesta, joka olisi vain sivuston omistajan käytössä.

Koska tässä työssä tehtiin verkkokauppa pääasiassa vain Laravelin ominaisuuksien testaamista varten, niin ominaisuuksia pystyttiin rajaamaan pois. Tarkemmat tuotesivut ja maksamisprosessi rajattiin pois, eikä hallintasivuille ollut myöskään minkäänlaista tarvetta, mutta nekin toteutetaan mahdollisesti tulevaisuudessa opinnäytetyön jälkeen. Toteutettaviksi toiminnallisiksi pääominaisuuksiksi jäi kirjautuminen, rekisteröinti, salasanan palautus, ostoskori ja karusellivalikko mainoksia varten. Frontend-osaamisen näyttäminen ja uusien tekniikoiden käyttö sai myös suuren painon työn edistyessä.

7.1.2 Ei-toiminnalliset vaatimukset

Toiminnallisten vaatimusten lisäksi sivustolla on myös paljon ei-toiminnallisia vaatimuksia, jotka ovat aivan yhtä tärkeitä, elleivät jopa tärkeämpiä vaatimuksia. Tällaisia ei-toiminnallisia vaatimuksia voidaan määritellä melko vapaasti sivuston tarpeiden mukaan. Verkkokauppaa varten tärkeimpiä vaatimuksia on ainakin tietoturva, nopeus, vakaus ja helppokäyttöisyys. Verkkokaupan tietoturva on ehdottomasti tärkein vaatimus verkossa toimivalle yritykselle. Laravelissa tietoturva on hoidettu valmiiksi erityisen hyvin, ja se on yksi sen vahvuuksista. Kuitenkin esimerkiksi maksamisprosessin toteutuksessa siihen pitää kiinnittää erityistä huolellisuutta, koska Laravel ei tarjoa mitään integroitua maksamisominaisuutta. Onneksi maksamiseenkin on kuitenkin tarjolla valmiita paketteja joita voi käyttää helposti Laravelin ohella.

Sivuston nopeus on tietenkin toinen yleinen ei-toiminnallinen vaatimus, johon vaikuttavat monet seikat. Sivuston nopeuteen vaikuttaa muun muassa käytetyt kehykset, tekniikat, ohjelmointikielet, tietokannat ja mainosten käyttö. Muita ei-toiminnallisia vaatimuksia sivustolla on ainakin onnistunut sivun ulkoasu ja sen helppokäyttöisyys. Tässä työssä ei ollut oikeastaan tarvetta miettiä ei-toiminnallisia vaatimuksia kovinkaan paljoa, mutta erityisesti tietoturvaan, nopeuteen ja sivuston vakauteen keskityttiin koko suunnittelu- ja toteutusprosessin ajan.

Myös ulkoasun toimivuutta ja helppokäyttöisyyttä korostettiin koko työn ajan.

7.2 Suunnittelu

Työn suunnitteluun käytettiin lähinnä aikaisempia kokemuksia verkkosivujen suunnittelusta ja toteutuksesta. Aluksi mietittiin hiukan vaatimuksia ja käytettäviä ohjelmia. Tein myös alustavan suunnitelman ulkoasusta ihan vain lyijykynällä paperille piirtämällä ja sen pohjalta vielä Photoshop-ohjelmaa käyttäen alustavan layout-suunnitelman pienen luonnoksen miltä ulkoasu voisi näyttää. Osa alkuperäisistä ideoista jäi pois ja tilalle tuli pari korvaavaa ideaa.

7.2.1 Tietokannat

Työssä käytettiin sqlite-tietokantaa, koska se on kevyt ja nopea, joten se on juuri täydellinen pienen testiprojektin käyttötarpeita varten. Suuremman verkkokaupan tietokantana kannattaisi harkita kuitenkin jotain monipuolisempaa tietokantaa, kuten esim. MySQL-tietokantaa. Laravel tekee tietokantojen hallinnoimisesta helppoa. Laravelin artisanin-ominaisuuden avulla oli helppo lisätä ja muokata tietokantatauluja muun muassa käyttäjä- ja tuotetietojen varastoimista varten. Laravelin avulla oli myös helppo tehdä tarvittavia yhteyksiä tietokantojen välille tarvittaessa. Lopullisessa sivustossani ei kuitenkaan tarvittu kovinkaan monimutkaisia tietokantoja, vaikka aluksi suunniteltiinkin ja käytiin läpi monta opetusvideota ja tein niiden perusteella tietokantoihin liittyviä Laravel-harjoituksia.

7.3 Käyttöönotto

Aivan työn alussa oli hiukan vaikeuksia saada Laravel kokonaisuutena asennettua, koska ohjeet olivat yleisesti keskittyneet muuhun kuin Windows-ympäristössä työskentelyyn. Aluksi kaikki löydetyt esimerkit olivat suunnattu toisille käyttöjärjestelmille kuten OS-X tai Linux-pohjaisiin käyttöjärjestelmiin. En ollut kuitenkaan täysin vakuuttunut, että minun kannattaisi vaihtaa käyttöjärjestelmää vain Laravelin takia, vaikka harkitsin sitäkin vaihtoehtoa tosissani. Kokeilin melko monenlaista ratkaisua, mutta jokaisessa ratkaisussa oli jokin ongelma jota en saanut helposti selvitettyä. Hetken kokeilun jälkeen olin jo lähes vakuuttunut, että koko Lara-

velin asennus kannattaisi hoitaa virtuaalikonetta käyttämällä esim. Laravelin virallista Vagrant-pakettia, Homesteadia, koska sillä tavoin kaikki ympäristöön liittyvät toimintatavat olisivat juuri samoja kuin muitakin käyttöjärjestelmiä käyttäessä. Sitä ennen päädyin kuitenkin kokeilemaan vielä Laragon-nimistä uutta ohjelmaa, eli ohjelmaa jolla Laravelin sai asennettua nopeasti Windows-ympäristössä. Laragon oli niin hyvin toteutettu ja kätevä, että minun ei enää tarvinnut tutustua muihin vaihtoehtoihin.

Laragonin avulla tyhjä Laravel-projekti oli helppo luoda, mutta Laragonin mukana tuli myös Laravelin vaatima Apache-serveri, MySQL-serveri ja PHP-serveri, eli kaikki mitä tarvitsee Laravelin kehittämistyön aloittamiseen. Se ei myöskään vaikuta käyttöjärjestelmän toimintaan millään tavalla. Sen avulla sivuja oli myös helppo esikatsella, koska Laragonin käynnistämisen yhteydessä se loi myös toimivan osoitteen projektillesi kehitystyötä varten. Laravel oli siis erittäin helppo asentaa Laragonin avulla eikä ongelmatilanteita syntynyt ollenkaan vaan sain onnistuneen asennuksen heti ensimmäisellä yrityskerralla. Myös vanhemman Laravel versio 4:n käyttöönotto onnistuu Laragonilla. Yksi Laragonin hyvä ominaisuus on, että sen voi asentaa tarvittaessa vaikka muistitikulle, eli sama asennus toimii tarvittaessa myös monessa paikassa ja monella koneella työskenteleville. Laragon tarjoaa myös monipuolisemman ja kehittyneemmän komentorivin kuin Windows-käyttöjärjestelmän vakiokomentorivi.

7.4 Ulkoasu

Tässä osiossa käydään läpi pääasiassa sivuston frontend- eli asiakaspuolen osaa. Suuri osa sivuston frontend-näkyvyydestä jää kuitenkin esittämättä tässä osiossa, ja sen esittely jatkuu siis myös myöhemmin toiminnallisuuksia esitellessä.

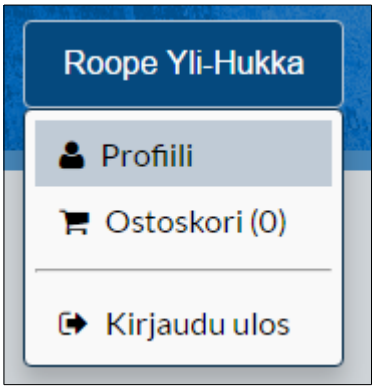
7.4.1 Fontit

Työssä käytettiin aluksi monia eri fontteja, mutta loppujen lopuksi siinä käytettiin vain ”Lato”-nimistä fonttia. Fontin on suunnitellut puolalainen suunnittelija nimeltään Łukasz Dziedzic. Fontin nimi on myös puolaa, ja se tarkoittaa suomeksi kesää. Työn logossa käytettiin kuiten-

kin muusta sivusta poiketen fonttia nimeltä ”Adobe Arabic Bold”.

7.4.2 Ikonit

Usein verkkosivuilla tulee tarvetta erilaisille ikoneille muun muassa käyttökokemuksen parantamiseksi. Ikoneita voi tarvita vaikka kirjautumispainiketta varten, tai ihan vain eri linkkien käyttötarkoituksen havainnollistamiseen. Työssäni oli myös tarvetta ikoneille, ja vaikka olen aiemmin tehnyt tarvittavat ikonit itse, niin hetken tutkittuani havaitsin, että on helpompikin tapa. Käytin siis työssäni ensimmäistä kertaa alun perin Twitterin bootstrap-kehystä varten suunniteltua ”Font Awesome”-työkalua. Sen avulla ikoneita pystyy käyttämään ikään kuin fontteina, jolloin niille saa myös samat CSS-ominaisuudet, kuin tavalliselle tekstillekin. Sen avulla ikoneista saa juuri sopivan värisiä ja kokoisia ja niille voi CSS:n avulla määrittää kätevästi vaikka omat varjonsa. Seuraavaksi on esimerkki ”Font Awesome”-ikonien käytöstä pudotusvalikossa (KUVIO 15).



KUVIO 15: "Font Awesome"-ikonien käyttö

7.4.3 Logo

Halusin verkkokaupalle yksinkertaisen, mutta samalla tyylikkään logon joka olisi tarvittaessa muokattavissa eri värisiin teemoihin tai vaikka eri vuodenaikoihin tai tiettyjä sesonkeja varten. Ulkoasusta oli monta eri versiota, eli myös logo vaihtui ulkoasun mukana pari kertaa. Tein lopullisen logon itse Photoshop-kuvankäsittelyohjelmaa käyttäen. Logossa käytetty fontti on nimeltään ”Adobe Arabic Bold”. Lopullisessa logossa on käytetty väreinä sinistä, mustaa ja val-

koista, eli samoja värejä kuin sivustossa pääasiassa muutenkin. Alla on lopullinen logo vaaleammalla pohjalla (KUVIO 16).

KUVIO 16: Saali's verkkokaupan logo.

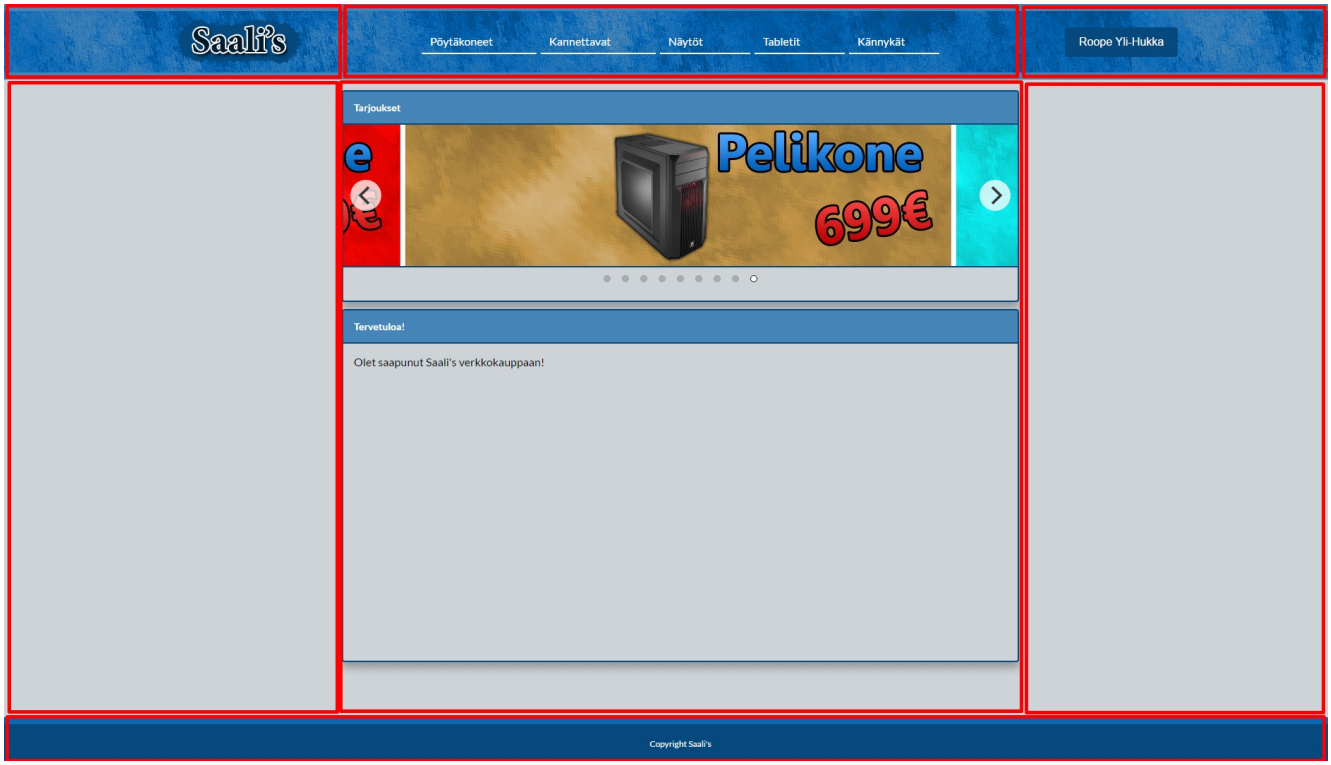
7.4.4 Tyylitiedosto

Aivan projektin alussa halusin testata Twitterin kehittämää bootstrapia layoutin rakentamiseen, mutta totesin sen olevan vastoin omia design-periaatteitani, koska haluan rakentaa mahdollisimman paljon sivustoni ulkonäöstä itse sekä tehdä sen juuri sillä tavalla, millä itse haluan. Vaikka sivuston olisi voinut toteuttaa pitkälti valmiilla bootstrap-tyyleillä, niin tein silti sivuston ulkoasun kokonaan itse, eli kaikki CSS-koodit on itse kirjoitettua. Sanoisin, että front-end on erityisosaamistani ja haluankin siihen siksi panostaa ja oppia uutta. Uusia CSS-tekniikoita tuli käytettyä testisivua rakentaessa, muun muassa ”flexbox”-ominaisuutta harjoiteltiin ja käytettiin paljon tämän testisivuston luomisessa. Lyhyesti flexbox, pidemmin ”CSS Flexible Box Layout Module”, on CSS3 version myötä tullut suosituksi tavaksi rakentaa sivuston layout, koska sen avulla elementit saa helpommin asetettua ja muutenkin sen käyttö on helpompaa kuin aikaisemmat vaihtoehtoiset vaihtoehdot. Pituutta CSS-tiedostolle tuli jopa 462 riviä koodia, koska se oli melko huonosti järjestetty ja jopa sotkuinen. Uskonkin, että saisin CSS-tiedostosta paljon siistimmän ja pienemmän, jos olisin suunnitellut sen alusta asti paremmin enkä vain lisännyt pikkuhiljaa lisää ja lisää koodia. Tätä testiprojektia varten siihen ei kuitenkaan ollut erityistä tarvetta.

7.4.5 Ulkoasun rakenne ja käyttäminen

Sivustoni ulkoasu on jaettu seitsemään eri osaan. Yläosassa on kolme erikokoista osaa: yksi

logolle, toinen linkeille ja kolmas kirjautumista ja rekisteröitymistä varten sekä kirjautumisen jälkeen myös muita ominaisuuksia varten, kuten esim. ostoskoria varten. Myös kielivalinnat sijoitettaisiin ylä-oikealle. Keskiosa jakautuu myös kolmeen osaan, joista keskimmäinen on pääsivu ja sivulla olevat toimivat tilana esim. mainoksille tai lisävalikoille kuten esim. tuotteiden hakemiseen. Lopuksi alimpana on vielä osio sivuston alatunnisteelle. Seuraavaksi osiot on vielä havainnollistettu punaisella värillä (KUVIO 17).



KUVIO 17: Ulkoasun rakenne

7.4.6 Verkkokaupan nimi

Vaikka verkkokauppa tehtiin vain testikäyttöön ja siitä puuttui toiminnallisuuksia, niin se sai silti nimen, joka voisi mahdollisesti vedota suomalaisiin. Verkkokaupan nimeksi valikoitui loppujen lopuksi Saali's joka viittaa siis hyvään saaliiseen, mutta se viittaa samalla myös, että verkkokaupan omistaja on nimeltään, tai tässä tapauksessa omistajan nimimerkki on ”Saali”. Idea tuli suositusta verkkokaupasta ”Jimm's.fi”.

7.5 Toiminnallisuus

Tässä osassa käydään läpi sivuston lopullista toteutusta ja sen toiminnallisuutta. Ratkaisuja havainnollistetaan sekä kuvilla että myös lyhyillä koodipätkilläkin.

7.5.1 Navigointi

Sivulla navigointi tapahtuu pääasiassa päälinkkejä painamalla, jotka on sijoitettu sivuston ylä-osaan. Kirjautumis- ja rekisteröintipainikkeet ovat esillä jos kukaan ei ole kirjautuneena, mutta niiden tilalla on liukuvalikko jos käyttäjä on kirjautuneena. Liukuvalikosta löytyy linkki profiiliin ja ostoskoriin, sekä uloskirjautumiseen. Seuraavaksi kuva navigaatiopalkista (KUVIO 18) ja heti perään pätkä koodia jolla se on toteutettu (KUVIO 19).

KUVIO 18: Navigointi

```
    <div class="navigointi-keski">
    <a class="keski2" href="{{ url('/poytakoneet') }}">Pöytäkoneet</a>
    <a class="keski2" href="{{ url('/kannettavat') }}">Kannettavat</a>
    <a class="keski1" href="{{ url('/naytot') }}">Näytöt</a>
    <a class="keski1" href="{{ url('/tabletit') }}">Tabletit</a>
    <a class="keski1" href="{{ url('/kannykat') }}">Kännykät</a>

    </div>
<div class="navigointi-oikea">
@if (Auth::guest())
    <a class="kirjautuminen"href="{{ url('/login') }}">Kirjautuminen   <i class="fa fa-btn fa-sign-out"></i></a>
    <a class="kirjautuminen"href="{{ url('/register') }}">Rekisteröinti   <i class="fa fa-btn fa-user-plus"></i></a>
</div>
@else

<div class="dropdown">
    <button class="dropbtn">{{ Auth::user()->name }}</button>
    <div class="dropdown-content">
        <a href="#"><i class="fa fa-btn fa-user"></i>Profiili</a>
        <a href="/cart"><i class="fa fa-btn fa-shopping-cart"></i>Ostoskori ({{Cart::count()}})</a>
        <hr/>
        <a href="{{ url('/logout') }}"><i class="fa fa-btn fa-sign-out"></i>Kirjaudu ulos</a>
    </div>
</div>
</div>
@endif
```

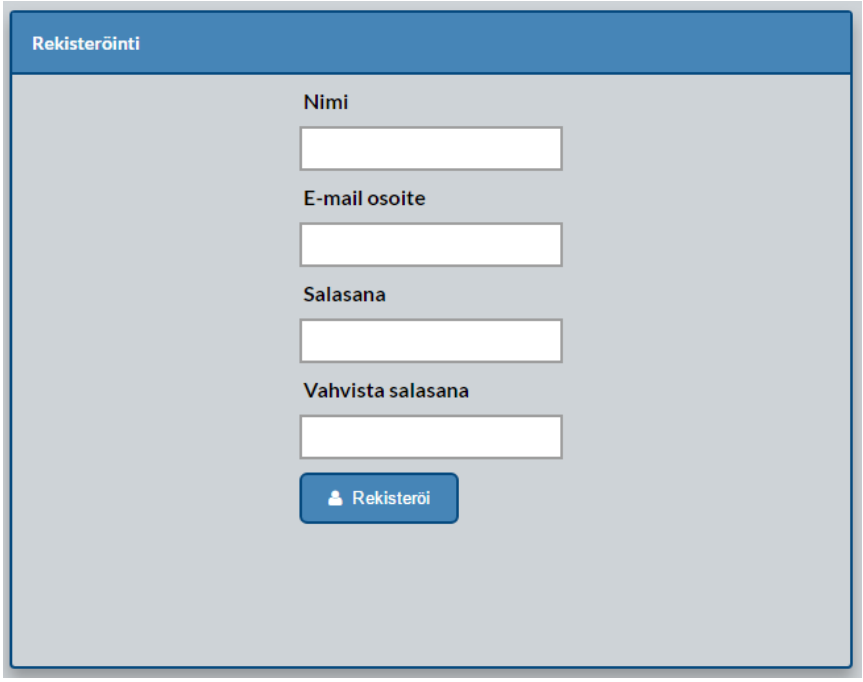
KUVIO 19: Osa navigointilinkkien koodista

7.5.2 Rekisteröinti, kirjautuminen ja salasanan palautus

Laravelin autentikointi (Engl. authentication) eli tunnistaumisominaisuus on todella hyvin toteutettu. Sen avulla rekisteröitymisen ja kirjautumisominaisuuden toteuttaminen on todella helppoa. Uuden Laravel-projektin asentamisen jälkeen komentoriviin syötetään vain komento ”php artisan make:auth” ja Laravel tekee lähes kaiken valmiiksi puolestasi. Tietokanta pitää luoda itse tai käyttää jotain jo luotua, kuten esim. Laragonin tarjoamaa MySQL-tietokantaa. Käytin kuitenkin itse sqlite-tietokantaa, joten loin sen ”database”-kansioon ja nimesin sen ”database.sqlite”. MySQL on määritetty ”database”-konfiguraatiotiedostoon oletustietokannaksi, mutta minun tapauksessani piti muuttaa oletustietokannaksi sqlite, joka kävi helposti. Tietokantaan piti vielä tuoda tarvittavat aiemmin määritetyt taulut käyttäjiä varten artisan-komennolla ”php artisan migrate”. Siinä oli kaikki, mitä Laravelin kirjautumisominaisuuden käyttöönotto minimissään vaatii. Kaikki tämä kannattaa tehdä heti projektin alussa, koska reititystiedostoon tulee muutoksia, jotka voisivat koitua ongelmallisiksi myöhemmässä vaiheessa projektia.

Kaikille käyttäjän tunnistautumiseen luodut toiminnot saavat myös automaattisesti generoidut ulkoasut ja jopa melko monimutkaiset ja perusteelliset reititysominaisuudet. Alkuperäinen ulkoasu on toteutettu Twitterin bootstrap-kehyksen päälle, mutta sitä voi soveltaa vapaasti ja tarvittaessa voidaan tehdä myös täysin oma ulkoasu. Tässä työssä on käytetty täysin omaa ulkoasua, joten myös kirjautumisen, rekisteröinnin ja salasanan palautuksien ulkoasut ovat kaikki täysin omaa koodia. Myös muita pieniä muutoksia tarvittiin esim. reititysten parantamiseen projektiani varten. Näiden ominaisuuksien automaattisesti generoituja koodeja oli ripoteltu vähän sinne ja tänne Laravelin hakemistoihin, joten minulla meinasi välillä olla vaikeuksia löytää paikkoja, missä tietyn toiminnallisuuden antava koodinpätkä oli, jos halusin vaikkapa muokata näitä toiminnallisuuksia. Lopuksi sain kuitenkin kaiken toimimaan juuri kuten halusinkin.

Rekisteröinti, kirjautuminen ja salasanan palautus saatiin kaikki toimimaan ja näyttämään sivustoon sopivalla tavalla. Näkymistä saatiin selkeitä, ja virheilmoitukset saatiin näkymään haluamallani tavalla. Seuraavassa kuvassa on verkkokaupan rekisteröitymisnäkymä (KUVIO 20) ja sitä seuraavassa kuvassa hiukan rekisteröitymisen koodia (KUVIO 21).



KUVIO 20: Rekisteröintinäkymä

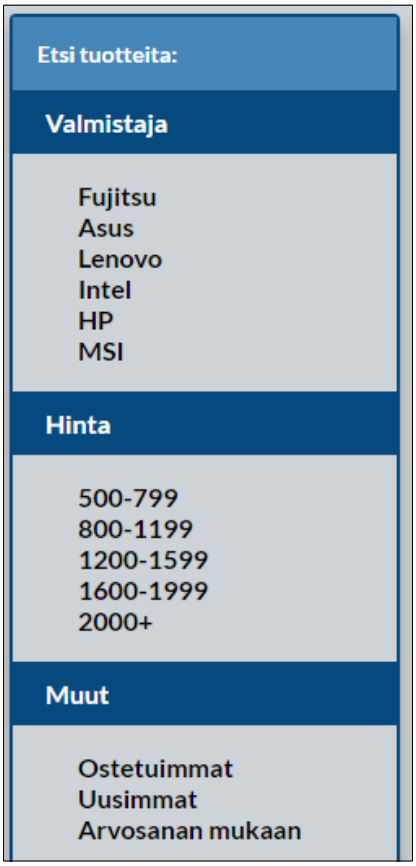
```
@section('content')
    <div class="keski-login">
                        @if ($errors->has('name'))
                            <span class="error">
                                <strong>Nimi kentässä on virhe.</strong>
                            </span>
                        @endif
                        @if ($errors->has('email'))
                            <span class="error">
                                <strong>Sähköposti kentässä on virhe.</strong>
                            </span>
                        @endif
                        @if ($errors->has('password'))
                            <span class="error">
                                <strong>Salasana kentässä on virhe.</strong>
                            </span>
                        @endif
                        @if ($errors->has('password_confirmation'))
                            <span class="error">
                                <strong>Salasanan vahvistus ei täsmää.</strong>
                            </span>
                        @endif
                    <div class="box">
                    <div class="box-otsikko"><p class="otsikko">Rekisteröinti</p></div>
                    <div class="box-rivi"></div>
                    <div class="login">
                    <form class="form-horizontal" role="form" method="POST" action="{{ url('/register') }}">
                        {!! csrf_field() !!}
                            <label class="otsikko">Nimi</label>
                                <input type="text" class="form" name="name" value="{{ old('name') }}">
                            <label class="otsikko">E-mail osoite</label>
```

KUVIO 21: Osa rekisteröintinäkymän koodista

7.5.5 Tuotteet

Lopullisen verkkokaupan olisi tarkoitus myydä valikoima laadukkaita valmiita pöytäkone ja kannettavia konepaketteja, mutta myös tabletteja ja kännyköitä. Työtä varten ei kuitenkaan

tehty erillisiä sivuja yksittäisille tuotteille, koska tuotteita käytettiin vain testikäytössä ostoskoria varten, eli niitäkään ei ollut montaa. Oikeaa verkkokauppaa varten tuotteille pitäisi kuitenkin tehdä oma taulu tietokantoihin, mutta Laravelin ansiosta sekin on helppoa ja nopeaa. Täydellisessä verkkokaupassa tuotteet jaettaisiin pääryhmiin ja pääryhmien sisällä niitä voidaan etsiä vaikka merkin tai hinnan mukaan. Toteutin valikon tuotteiden etsimiselle. Etsintävalikko sijaitsee tuotesivun vasemmalla puolella. Alla olevassa kuvassa on etsintävalikko ”Pöytäkoneet” pääryhmälle (KUVIO 22).

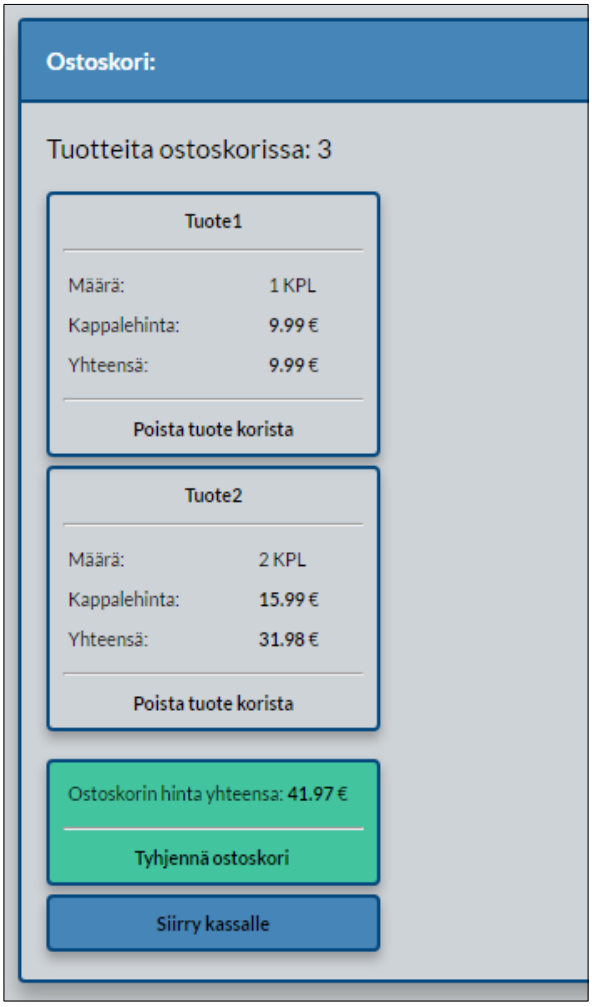


KUVIO 22: Etsintävalikko pöytäkoneille

7.5.6 Ostoskori

Ostoskori-sivu näyttää kaikki sinne lisätyt tuotteet tietoineen ja laskee myös lopullisen hinnan. Jos samaa tuotetta oli monta kappaletta, näytettiin ne yhtenä tuotteena, mutta määrä ja hinta kasvoi lukumäärän mukaan. Ostoskorin voi tyhjentää painamalla ”Tyhjennä ostoskori”-nap-

pia. Sivuston toiminnallisuutta jouduttiin rajaamaan hiukan, joten tuotteiden tilaamista ja maksutapahtumaa ei toteutettu tässä työssä loppuun asti, koska sellaiset ominaisuudet ovat melko vaikeita ja työläitä toteuttaa eikä niille ollut mitään oikeaa tarvetta. Tarkoituksena olisi kuitenkin jatkaa työtä jälkikäteen ja toteuttaa myös tuotteiden maksaminen. Alla ostoskori-näkymästä parin lisätyn tuotteen jälkeen (KUVIO 23).



KUVIO 23: Ostoskori-sivu

Alla olevassa kuvassa on pätkä ostoskori-sivuston koodista (KUVIO 24). Koodi havainnollistaa hyvin, kuinka blade-sivumoottori selventää koodia. Joka väliin ei tarvitse laittaa PHP-kielen perinteisesti vaatimaa ”<?php”-tägiä toistuvasti puhumattakaan tulostuksesta, eli toistuvasta ”echo”, vaan voidaan ilmoittaa PHP-tulostuksesta lyhyesti kahta aaltosulkua käyttämällä. Tämä säästää aikaa, mutta myös selventää koodia mukavasti. Nähdään myös, että silmukan käyttö on hiukan siistimpää blade-moottoriin kuuluvan ”@”-merkinnän avulla. Kuvassakin

näkyvä ”foreach”-toistosilmukka aloitetaan ”@foreach” ja lopetetaan ”@endforeach”.

```
                    <p class="boxi">Tuotteita ostoskorissa: {{ Cart::count() }} </p>
    @foreach($cart as $row)
<table class="kori">
        <tr class="kori">
            <td class="kori-keskitys" colspan="2"><strong>{{ $row->name }}</strong></td></tr>
            <tr><td colspan="2"><hr></td></tr>
          <tr class="kori"><td class="kori">Määrä:</td> <td class="kori">{{ $row->qty }} KPL</td></tr>
          <tr class="kori"><td class="kori">Kappalehinta:</td><td class="kori"><b>{{ $row->price }} </b> €</td></tr>
          <tr class="kori"><td class="kori">Yhteensä:</td> <td class="kori"><b>{{ $row->subtotal }} </b> €</td></tr>
          <tr><td colspan="2"><hr></td></tr>
          <tr class="kori"><td class="kori-keskitys" colspan="2">
          <a class="kori-poista" href="/cart/remove/{{ $row->id }}">Poista tuote korista</a></td></tr>
      </tr>
</table>
    @endforeach

    <p class="boxi">
    <table class="kori2">
    <tr class="kori"><td class="kori">Ostoskorin hinta yhteensa: <b> {{ Cart::total() }} </b> € </td></tr>
    <tr><td colspan="2"><hr></td></tr>
    <tr class="kori"><td class="kori-keskitys"><a class="kori-poista" href="/clearcart">Tyhjennä ostoskori</a> </td></tr>
    </table>
```

KUVIO 24: Osa ostoskori-sivun koodista

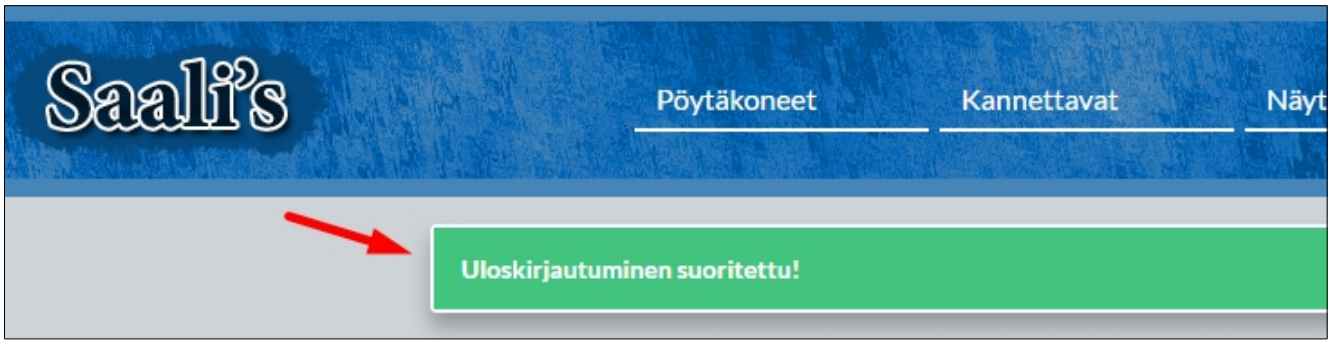
7.5.7 Karusellivalikko

En ollut aikaisemmin toteuttanut minkäänlaista karusellivalikkoa, joten halusin oppia toteuttaa sellaisen testisivustolleni kokemuksen karttamiseksi. Minulla ei ollut aluksi aavistustakaan mistä aloittaa, joten tutkin erilaisia saatavilla olevia valmiita ratkaisuja melko laajasti. Osa toimivasta ratkaisuista oli ikävä kyllä maksullisia käyttää, joten päädyin tekemään karusellin ”Flickity”-nimisen ilmaisen CSS-kehyksen pohjalle. Flickity käyttää CSS:n lisäksi myös JavaScript-ohjelmointikieltä dynaamisen toiminnallisuuden saavuttamiseksi. Sen käyttö oli dokumentoitu erinomaisesti ja se olikin pääsyy miksi sen valitsin. Se oli helposti käytettävä, mutta se oli myös monipuolinen ja todella joustavasti muokattavissa. Tein sitä varten pari mainoskuvaa sen käytön testaamiseksi ja lopputulos näytti tältä (KUVIO 25).

KUVIO 25: Flickityllä toteutettu karusellivalikko

7.5.8 Ilmoitukset (Virheilmoitukset ja flash viestit)

Sivuston käyttäjälle näytettävät viestit ovat melko tärkeä toiminnallisuus, koska tällä tavoin käyttäjälle voidaan näyttää, mitä sivuston palvelinpuolella tapahtuu. Sivustossa voi normaalisti tapahtua kaikenlaista ilman, että käyttäjä saa siitä minkäänlaista näkyvää tietoa, joten viestit ovat oiva tapa edistää sivuston käyttäjäkokemusta. Koska halusin toteuttaa sivuston suomenkielisenä, niin minun piti muuttaa hiukan Laravelin valmiiksi generoituja virheilmoitusviestejä, joita sivusto näytti muun muassa epäonnistuneissa kirjautumis- tai rekisteröitymisyrityksissä. Laravelin avulla virheilmoituksista sai monipuolisia ja niitä oli suhteellisen helppo automatisoida käyttämällä tarjolla olevia sääntöjä. Laravelin reitityksen avulla sai tehtyä monipuolisesti vaikka mitä, mutta yksi helpoimmin toteutetuista ominaisuuksista, oli niin sanottujen flash-viestien käyttäminen. Alla esimerkki erikseen itse toteutetusta flash-viestistä Laravelin reititystä hyödyksi käyttäen (KUVIO 26) ja heti sen alla samaa esimerkkiä varten kirjoitetusta koodista (KUVIO 27).

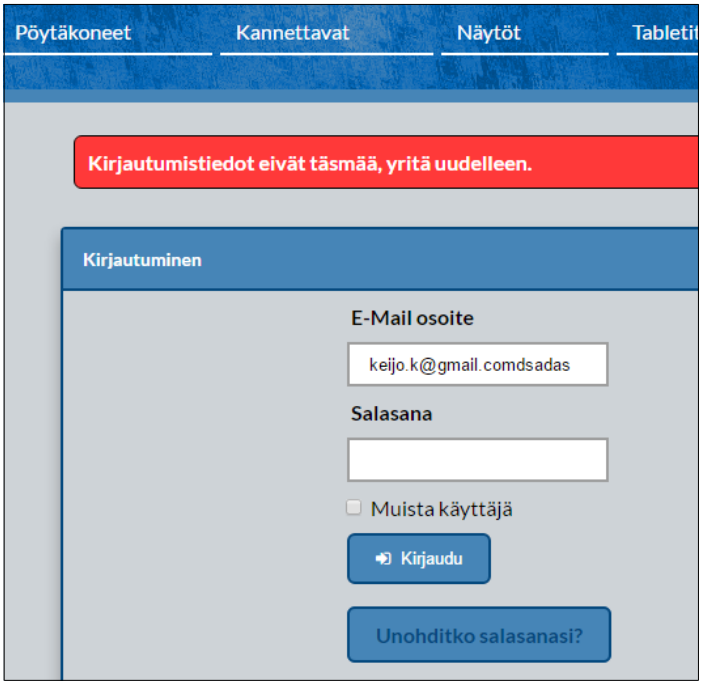


KUVIO 26: Uloskirjautumisen jälkeinen flash-viesti

```
Route::get('loggedout', function () {
    Session::flash('flash_message', 'Uloskirjautuminen suoritettu!');
    return redirect('/');
});
```

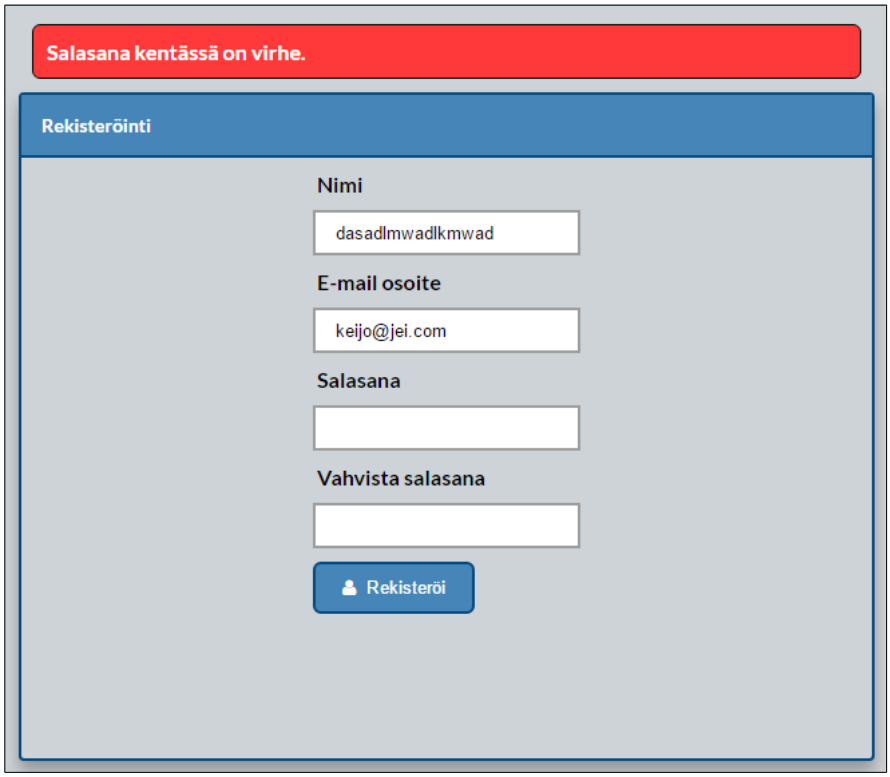
KUVIO 27: Reitityksellä toteutettu flash-viesti

Virheellisiä kirjautumisen virheilmoituksia oli monta ja nekin olisi tarvittaessa voitu kääntää samanaikaisesti monelle kielelle. Sille ei kuitenkaan ollut tarvetta testikäyttöön tehdyssä sivustossa. Alla on esimerkki virheilmoituksesta (KUVIO 28).



KUVIO 28: Kirjautumisen virheilmoitus

Myös rekisteröintiä varten tarvittiin vastaavat omat virheilmoitukset. Laravel generoi myös automaattisesti sähköpostiosoitteen tarkistuksen, joka oli toteutettu JavaScriptillä, eli se oli hiukan dynaamisempi kuin pelkästään PHP-kielellä toteutettu tarkistus. Virheellisen e-mail-osoitteen syötön jälkeen sitä koskevat virheilmoitukset tulivat heti tiedon syötön jälkeen ilman, että sivua tarvitsi tai edes ehdittiin päivittämään. En kuitenkaan selvittänyt JavaScriptillä toteutettua koodia sen tarkemmin. Rekisteröimisen epäonnistumisen yhteydessä annettu virheilmoitus alla olevassa kuvassa (KUVIO 29).



KUVIO 29: Virheilmoitus rekisteröimisen yhteydessä

7.6 Testaus

Vaikka sivustoa ei tehty tilaustyönä kenellekään varsinaiselle asiakkaalle, vaan opittujen asioiden testaamiseen, testaaminen oli silti iso osa projektia. Testaus tapahtui yleensä heti lisätyn toiminnon tai asian jälkeen, eli mitään suurempaa erityistä testaussessiota ei loppujen lopuksi tarvittu. Suurin osa testaamisesta oli opittujen ominaisuuksien toimivuuden tavoittelua. Testaamiseen kuului muun muassa kirjautumisen, rekisteröitymisen ja salasanan palautusominaisuuksien toimivuuteen liittyviä. Sen lisäksi muun muassa ulkoasuun liittyvät seikat olivat melko suuressa osassa itse testausprosessia.

7.7 Ongelmatilanteet

Ongelmatilanteet ovat osa ohjelmistokehittäjän arkea, eli myöskään tässä työssä ei niiltä vältytty. Ongelmatilanteet ovat epäilemättä kehittämistyön yksi kiehtovin ja koukuttavin puoli, mutta ne ovat usein myös sen ärsyttävin ja turhauttavin puoli. Suurempia mainitsemisenarvoisia ongelmatilanteita, jotka olisivat saaneet projektin pysähtymään pidemmäksi aikaa, ei kuitenkaan oikeastaan ollut. Pienempiä ongelmatilanteita oli kuitenkin melko paljon.

Aivan aluksi heti kehyksen valitsemisen jälkeen ympäristön valitseminen tuotti vaikeuksia, mutta onneksi siihenkin löytyi ratkaisu, joka oli uskoakseni myös henkilökohtaisesti paras mahdollinen. Aloitin Laravelin opiskelun opetusvideoita katsomalla ja minun oli aluksi vaikea saada Windows-käyttöjärjestelmään opetusvideoiden tekijän kanssa samankaltaista kehitysympäristöä, koska niiden tekijä käytti Applen kehittämää OS X-käyttöjärjestelmää Windowsin sijaan. Lyhyen tiedustelun jälkeen sain käsityksen, että yleinen mielipide parhaasta ympäristöstä olisi jokin Linux-pohjainen käyttöjärjestelmä. Vaikka minulla oli hiukan kokemusta Linux-ympäristöstä, niin oli kuitenkin suhteellinen iso kynnys aloittaa kehitys sillä eikä minulle tutummassa Windows-ympäristössä. Windows-ympäristössä Laravelin kehittäminen oli paljon mutkikkaampaa, kunnes löysin Laragon-ohjelmiston jonka avulla asennus ja käyttöönotto oli todella vaivatonta ja jopa helppoa.

Myös Laravelin käytännön testauksessa testisivustoa tehdessä tuli paljon tilanteita, joita en osannut odottaa enkä siten aina löytänyt heti ratkaisua. Lukemattomia kertoja tapahtui sellaisia ongelmia, joita ei mielestäni pitänyt voida tapahtua, mutta aina lopulta löysin ongelman aiheuttajan ja tavan millä ne pystyi kiertämään tai korjata. Jos ongelmaan ei löytynyt heti vastausta, oli helppo kysyä suoraan muilta käyttäjiltä, että olisiko heillä ratkaisua pulmaan. En joutunut kysymään apua paria kertaa useammin, mutta aina kun kysyin, niin sain vastauksen. Eli tästä voimme päätellä, että Laravelin käyttäjäkunta on hyvin aktiivinen ja heti valmiina auttamaan.

Halusin sivustani suomenkielisen ja huomasin virheilmoituksia kääntäessäni hiukan vaikeuksia saada virheilmoitukset oikeaan muotoon. Suomenkielessä sanat taipuivat välillä hiukan

vaikeasti, eli en sen takia voinut kääntää kaikkia englanninkielisiä virheilmoituksia suoraan suomenkielelle. Olisin tietenkin voinut vain todeta, että helpompi tehdä koko sivusto englanniksi, mutta halusin tutkia kieliominaisuuksia kokemuksen karttamiseksi ja ihan vain omasta mielenkiinnostakin aihetta kohtaan.

Yleensä syntyneet ongelmatilanteet olivat hyvinkin tapauskohtaisia, eli välillä meinasi olla ongelmia paikantaa ongelman aiheuttaja tai lähde. Kokeneemmat auttajat osasivat kuitenkin usein vihjata mistä etsiä ja ongelmat ratkesivat sen avulla melko helposti paria aikaa vievää poikkeusta lukuun ottamatta. Suomessa Laravelin käyttö on vielä suhteellisen vähäistä, ja välillä olisinkin toivonut saavani mahdollisuuden myös avun saamiseen englannin lisäksi myös suomeksi.

8 YHTEENVETO

Laravelin loistavat ominaisuudet ja suhteellisen käyttäjäystävällinen oppimisprosessi on saanut Laravelin saavuttamaan suuren suosion kohtuullisen pienessä ajassa, eli aivan syyttä suotta Laravelia ei ole kehuttu yleisesti maasta taivaisiin lähes jokaisessa keskustelussa mitä aiheesta löytää. Laravelin valinta ensimmäiseksi ohjelmistokehykseksenі oli mielestäni hyvin onnistunut ja suosittelenkin kaikkia www-ohjelmoinnista kiinnostuneita kokeilemaan Laravelia tai jotain muuta samankaltaista ohjelmistokehystä. Kehyksen ominaisuuksien ja käytön opiskelu on toki aikaa vievää puuhaa, mutta myös hyvin palkitsevaa.

Kun ohjelmistokehyksen keskeisimmät seikat on tarpeeksi hyvin ymmärretty ja sen ominaisuuksien käyttöä muutaman kerran kokeillut, niin oikein käytettynä kehyksen käyttö nopeuttaa ja tehostaa varmasti kaikenkokoisia projekteja. Kehyksen ansiosta ei tarvitse enää keksiä ratkaisuja ja rakentaa kaikkea aina uudelleen itse vaan saa käyttöön suuren määrän toimivia ratkaisuja, joita käyttää ja mukauttaa tarvittaessa tarpeidesi mukaan. Käyttäjällä on myös täysi vapaus tehdä muutoksia ja muuttaa jotain tiettyä tapaa jolla ongelmat on ratkaistu, mutta kokemukseni perusteella ne on yleensä hoidettu todella hyvin jo valmiiksi. Haluan vielä korostaa, että Laracasts.com-sivuston laaja opetusvideovalikoima helpotti todella paljon omaa opiskeluani ja sai samalla pidettyä mielenkiinnon yllä, eli kehotankin kiinnostuneita kurkkaamaan vielä sivuston tarjontaa.

Laracasts-opetussivusto oli siis henkilökohtaisesti juuri se syy, miksi innostuin koko ohjelmistokehyksestä, koska vaikeimpienkin käsitteiden ja asioiden oppiminen oli tehty paljon selvemmäksi opetusvideoiden avulla. Sivuston opetusvideoiden laajuus sai minut janoamaan enemmän ja enemmän myös sellaista tietoa, jota tuskin tulen edes vielä hetkeen tarvitsemaan omissa projekteissani.

8.1 Toteutuksen arviointi

Olen hyvin tyytyväinen Laravelilla saamiini tuloksiin ja vaikka matkalla olikin mutkia ja ongel-

mia niin löysin lopuksi aina toimivan ratkaisun ja sain tehtyä kaiken haluamallani tavalla ilman, että jouduin tyytymään kompromisseihin. Sain Laravelin avulla sivustooni paljon sellaista toiminnallisuutta, jota en ole aiemmin kyennyt toteuttamaan. Uskon myös, että Laravelin käyttö parantaa tekemieni sivustojen tietoturvaa huomattavasti, koska Laraveliin integroidut tietoturvaratkaisut ovat aivan eri luokkaa kuin aiemmin itse käyttämäni ratkaisut. Laravel tekee automaattisesti myös paljon sellaista, mitä ei edes huomaa sen tekevän. Vaikka kehyksen soveltaminen projektissa oli aluksi melko hidasta ja työlästä, niin uskon koko suunnitteluprosessin nopeutuvan ajan myötä, kunhan tietyistä aluksi miettimistä ja keskittymistä vaativista asioista tulee rutiininomaisempaa.

8.2 Hankitun osaamisen arviointi

Vaikka minulla oli heti alussa selvä suunta ja selvät tavoitteet työni etenemisestä

niin kuitenkin kehyksen valinnan, opiskelun, käyttöönoton ja testauksen aikana syntyi myös paljon uusia näkökulmia, ideoita ja käytännön ratkaisuja, jotka eivät aikaisemmin olisi tulleet mieleenikään. Nyt voin sanoa osaavani soveltaa Laravelia projekteissani ja ymmärrän kehyksen käytön hyötyjä ja vahvuuksia, sekä osaan soveltaa oppimaani myös muiden alustojen opettelussa. Voin nyt siis sanoa osaavani Laravel-kehyksen perusteet. Myös MVC-arkkitehtuuri tuli ensimmäistä kertaa tutuksi ja ymmärrän hyvin, miksi se on niin suosittu. Pienemmissä ja vähemmän toiminnallisuutta vaadittavissa projekteissa MVC tai muita vastaavia ohjelmointimalleja tarvitsee tuskin hyödyntää, mutta sen käyttöä on hyvä harjoittaa vähän isompia projekteja varten. Laravelista oppimani perusteella uskon myös pystyväni mukautumaan ja oppimaan jonkin uuden ohjelmistokehyksen perusteet nopeaa jo Laravelista oppimieni asioiden ansiosta.

9 POHDINTA

Minulla ei alun perin ollut edes täysin selvää kuvaa siitä, että mikä se PHP-kehys oikeastaan edes on, ja sitä lähdin tässä samalla selvittelemäänkin. Opinnäytetyöni taustalla oli lähinnä vain se tieto, että ohjelmistokehykset ovat nykyaikaa www-ohjelmoinnissa, ja sen takia halusin selvittää niistä enemmän. Aikaisemmin olin tehnyt www-sivustoja ilman mitään sisällönhallintatyökalua tai ohjelmistokehystä, joten halusin selvittää, voinko hyötyä jotenkin käyttämällä jotain kehystä erillisten projektieni runkona. Sainkin opinnäytetyöni aikana hyvän kuvan ohjelmistokehyksistä ja siitä, miksi niitä käytetään. Sen lisäksi päädyin Laravel testiprojektia rakentaessani tekemään myös paljon sellaista työtä, joka ei vaikuttanut varsinaisesti lopputulokseen, mutta jotka ovat kuitenkin tärkeitä seikkoja websovellusten kehittämistyössä. Käytin myös paljon ylimääräistä aikaa muun muassa ulkoasun kanssa värkkäämiseen ja sen todistaa muun muassa täysin oman tyylitiedostoni pituus, joka sekin oli loppujen lopuksi yli 400 riviä pitkä. Koodin määrä ja pituus voivat viitata tosin myös huonoon koodin optimointiin, mutta se taas ei ollut yksikään tavoitteistani.

Olen harrastanut www-sivujen suunnittelua ja toteuttamista jo yli 10 vuotta. Innostukseni ohjelmointia kohtaan lähti omasta kiinnostuksestani, ja otinkin asioista selvää itsekseni ja alun perin vain omaksi ilokseni. Melko vaatimattomien www-sivujen toteuttaminen itselleni, perhetuttaville tai kavereilleni sai minut vakuuttuneeksi, että haluan tehdä jotain ohjelmointiin liittyvää tulevaisuudessa myös työkseni. Siitä syystä pidän ohjelmoinnista ja haasteista mitä ohjelmointi tuo, mutta myös kaikki ulkoasun suunnitteluun liittyvät seikat, kuten front-end-ohjelmointi tai graafinen suunnittelu kiinnostaa todella paljon. Tulen hyödyntämään Laravelia tai jotain muuta PHP-kehystä tulevissa projekteissani.

Uskon vakaasti, että kokemus ja tietämys ihan mistä tahansa nykyaikaisesta kehyksestä tai muista sivujen suunnitteluun ja toteutukseen liittyvistä seikoista tulee auttamaan minua myös työllistymisessä omalle alalleni. Työllistyminen www-sivujen parissa on ollut haaveenani niin kauan kuin vain muistan, ja sama kiinnostus on pysynyt kaikki nämä vuodet. Toivottavasti siis Laravel saa myös Suomessa suuremman suosion ja pääsen hyödyntämään oppimaani mahdollisimman pian harrastukseni lisäksi myös työelämässä.

Olen mielestäni hyvinkin käytännönläheinen ihminen ja huomasinkin Laravelin opiskelun aikana menettäneeni usein keskittymisen tai jopa mielenkiinnon välillä kokonaan, koska en aina päässyt testaamaan opiskelemiani asioita käytännössä heti. Usein asioita piti opiskella ajatuksella useampaan kertaan ennen kuin ymmärsin asiat oikein. Koko kokonaisuuden opettelu oli aluksi tietyllä tapaa jopa pelottavaa. Moni asia tuntui aluksi myös monimutkaiselta ja jopa ylitsepääsemättömältä oppia, mutta yritin olla ajattelematta kaikkea sitä kokonaisuutena ja opiskella asioita aina yksi kerrallaan.

Asiat alkoivatkin pikkuhiljaa loksahtelemaan paikoilleen ja kokonaisuus hahmottumaan paremmin, mitä pidemmälle opiskelussani pääsin. Kaikki muuttui, kun pääsin kokeilemaan ja soveltamaan kaikkea oppimaani käytännössä. Innostuin ja sain mielenkiintoni huippuun heti kun pääsin aloittamaan Laravelin käytön käytännössä eli pääsin suunnittelemaan ja toteuttamaan omaa projektiani sillä. Loppujen lopuksi työ sujuikin kuin tanssi, ja välillä en hyvän vauhdin ja innostukseni löydettyäni suonut itselleni edes taukoja, vaan istuin tuntikausia täydessä työn touhussa. Sellaista innostusta olenkin jo pitkään kaivannut, ja olen varma, että Laravel tarjoaa samaa myös tulevaisuudessa.

LÄHTEET

Bari, A. & Syam, A. 2008. CakePHP Application Development. Saatavissa: http://it-ebooks.info/book/2489. Viitattu: 1.6.2016.

Barnes, E. 2016. Laravel Release Process. Saatavissa: https://laravel-news.com/2016/01/laravel-release-process. Viitattu: 25.5.2016.

Barnes, E.L. 2016. Laravel Release Process. Saatavissa: https://laravel-news.com/2016/01/laravel-release-process. Viitattu: 30.5.2016.

Bean, M. 2015. Laravel 5 Essentials. Saatavissa: http://it-ebooks.info/book/6194. Viitattu: 4.6.2016.

Echher, C. 2014. Professional Web Design : Techniques and Templates (5). Saatavissa: http://site.ebrary.com.ezproxy.centria.fi/lib/cop/detail.action?docID=10884365 Viitattu: 9.6.2016.

Heng, C. 2015. What is MySQL? What is a Database? What is SQL? Saatavissa: http://www.thesitewizard.com/faqs/what-is-mysql-database.shtml. Viitattu: 9.6.2016.

HTML.net. 2015. Free CSS Tutorial. Saatavissa: http://html.net/tutorials/css/lesson1.php. Viitattu: 5.6.2016.

Kitipova, I. 2015. Most popular PHP Frameworks 2015. Saatavissa: https://www.webhostface.com/blog/popular-php-frameworks-2015. Viitattu 29.5.2016.

Lecky-Thompson, E. & Nowicki, S. 2009. Professional PHP6. Saatavissa: http://it-ebooks.info/book/348. Viitattu: 2.6.2016.

Lecky-Thompson, G. 2005. Corporate Software Project Management. Saatavissa: http://site.ebrary.com.ezproxy.centria.fi/lib/cop/reader.action?docID=10078507. Viitattu: 1.6.2016.

Lie, H. 2005. Cascading Style Sheets. Saatavissa: http://people.opera.com/howcome/2006/phd. Viitattu: 10.6.2016.

Long, J. 2012. I don't speak your language: Frontend vs. Backend. Saatavissa: http://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend. Viitattu: 6.6.2016.

MDN. 2016. JavaScript guide. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction. Viitattu: 26.5.2016.

Ntchosting.com. 2016. Databases. Saatavissa: https://www.ntchosting.com/encyclopedia/databases/structured-query-language Viitattu: 7.6.2016.

Otwell, T. 2016a. Laravel.com. Laravel dokumentointi. Saatavissa: https://laravel.com/docs/master/structure Viitattu: 1.6.2016.

Otwell, T. 2016b. Laravel.com. Laravel dokumentointi. Saatavissa: https://laravel.com/docs/master/providers. Viitattu: 1.6.2016.

Raittila, A. 2016. Hakukoneoptimointi lyhyesti. Saatavissa: http://nettibisnes.info/hakukoneoptimointi. Viitattu: 9.6.2016.

Rouse, M. 2010. Learn IT: Software development. Saatavissa: http://whatis.techtarget.com/reference/Learn-IT-Software-development. Viitattu: 1.6.2016.

Shannon, R. 2012. What is HTML? Saatavissa: http://www.yourhtmlsource.com/starthere/whatishtml.html. Viitattu: 9.6.2016.

Skvorc, B. 2015. The Best PHP Framework for 2015: Sitepoint Survey Results. Saatavissa: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results. Viitattu 7.6.2016.

Stauffer, M. 2016. Laravel: Up & Running. Saatavissa: http://it-ebooks.info/book/7047. Viitattu: 2.6.2016.

Surguy, M. 2013. History of Laravel PHP framework, Eloquence emerging. Saatavissa: https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging. Viitattu: 6.6.2016.

Surya, S. 2015. CMS or PHP Framework: Which technology is better for my business? Saatavissa: http://www.ithands.com/blog/cms-or-php-framework-which-technology-is-better-for-my-business. Viitattu: 7.6.2016.

The PHP Group. 2016. History of PHP. Saatavissa: http://php.net/manual/en/history.php.php. Viitattu 6.6.2016.

W3C. 2016. HTML. Saatavissa: https://www.w3.org. Viitattu: 1.6.2016.

W3Schools.com. 2016a. JavaScript Introduction. Saatavissa: http://www.w3schools.com/js/js_intro.asp. Viitattu: 9.6.2016.

W3Schools.com. 2016b. PHP 5 Tutorial. Saatavissa: http://www.w3schools.com/php. Viitattu: 9.6.2016.

W3Schools.com. 2016c. SQL Tutorial. Saatavissa: http://www.w3schools.com/sql. Viitattu: 9.6.2016.

W3Schools.com. 2016d. JavaScript Introduction. Saatavissa: http://www.w3schools.com/sql. Viitattu: 9.6.2016.

Way, J. 2012. Why Laravel is Taking the PHP Community by Storm. Saatavissa: http://code.tutsplus.com/tutorials/why-laravel-is-taking-the-php-community-by-storm–pre-52639. Viitattu: 27.5.2016.

Way, J. 2014. Object-oriented bootcamp. Opetusvideo. Saatavissa: https://laracasts.com/series/object-oriented-bootcamp-in-php. Viitattu: 2.6.2016.

Webopas.net. 2013. Mikä on ohjelmistotuotanto? Saatavissa: http://www.webopas.net/otuotanto.html. Viitattu: 1.6.2016.