SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

LARAVEL 8 -SISÄLLÖNHAL-LINTAJÄRJESTELMÄ VERKKOSIVULLE

TEKIJÄ/T: Susanna Rinnevuori

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Susanna Rinnevuori	
Työn nimi Laravel 8 -sisällönhallintajärjestelmä verkkosivulle	
Päiväys 17.12.2020	Sivumäärä/Liitteet 32
Toimeksiantaja/Yhteistyökumppani(t) Traakkoni Oy	

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli myös suunnitella sekä toteuttaa lopputuotoksena sisällönhallintajärjestelmä verkkosivustolle, jolloin verkkosivuston omistaja(t) pystyvät hallitsemaan dynaamisesti sen sisältöjä itse ilman ohjelmistokehittäjän apua. Opinnäytetyön tavoitteena oli myös tutustua Laravel 8 ohjelmistokehykseen ja sen tekniikoihin ja kehittää aiemmin opittua.

Sisällönhallintajärjestelmä eli Content Management System (CMS) on järjestelmä, jonka avulla ylläpitäjä voi hallita jotain kokonaisuutta ja sen sisältöjä. Kuuluisimpia ylläpitojärjestelmiä ovat Wordpress, Joomla ja Drupal. Sisällönhallintajärjestelmän voi luoda myös esimerkiksi käyttämällä jotakin olemassa olevaa ohjelmistokehystä. Tämä opinnäytetyö on toteutettu käyttämällä PHP:n Laravel ohjelmistokehyksen versiota kahdeksan. Opinnäytetyön aikana tutustuttiin Laravelin dokumentaatioon. Kehitysympäristönä käytettiin WampServeriä, joka sisältää Apachen, MySQL:n sekä PHP:n asennuksen. Lisäksi opinnäytetytössä käytettiin tekniikkoina Bootstrapiä, CSS:ää, Javascriptiä ja JQueryä sekä Tailwind CSS:ää.

Lopputuotoksena syntyi sisällönhallintajärjestelmä Single Page Application tyyppiselle sivustolle. Sisällönhallintajärjestelmää voidaan hallita selaimen kautta sekä tietokoneelta käsin, että mobiililaitteilla. Tämän lisäksi oppiminen kehittyi web-sovellustekniikoiden saralla, varsinkin Laravel ohjelmistokehyksestä. Vaiheet dokumentoitiin ja ne käydään läpi tässä opinnäytetyössä.

Avainsanat

Laravel, PHP, sisällönhallintajärjestelmä, web-ohjelmointi, ohjelmistokehys, MVC-malli

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Susanna Rinnevuori	
Title of Thesis Laravel 8 CMS for a Website	
Date 17 December 2020	Pages/Appendices 32
Client Organisation /Partners Traakkoni Oy	

Abstract

The purpose of this thesis was to create a Content Management System (CMS) for a Single Page Application (SPA) website. The main goal was to create an access to the owners of the website to dynamically maintain the contents without the need of contacting the administrating web developer. The purpose of this thesis was also to learn the usage and techniques of Laravel 8 PHP Framework and to deepen the knowledge about previously learned techniques.

Content Management System (CMS) is a system that allows the administrator to have control over contents. The most known Content Management Systems at the time of writing are Wordpress, Joomla and Drupal. CMS can also be created by using an existing framework and in this thesis the system was created with Laravel Framework version 8. An important part of the thesis was to study the documentation to be able to create the system. The development environment had to be created and WampServer was installed locally for this purpose. WampServer includes an Apache HTTP server, MySQL database and PHP. Other techniques used in this project were CSS, Bootstrap, Javascript, JQuery and Tailwind.CSS.

The final outcome of the project was an easy to use Content Management System for a Single Page Application. The system can be accessed through a web browser of a computer or a mobile device. Knowledge about web developing and especially about the Laravel framework was increased during the process. Steps to reach the results are documented and presented in this thesis.

Keywords
Laravel, PHP, CMS, web development, framework, MVC-architecture

SISÄLTÖ

1 JOHDANTO

Sisällönhallintajärjestelmä on järjestelmä, joka koostuu sisällöstä, tietokannasta ja jonkinlaisesta käyttöliittymästä, jolla sisältöä hallitaan. Sisällönhallintajärjestelmien tavoitteena on dynaamisen sisällön luonti, joka tarkoittaa sitä, että sisältöä voidaan luoda, päivittää ja poistaa.

Tämän opinnäytetyön aihe on saatu Traakkoni Oy:ltä. Traakkoni Oy oli jättänyt hakemuksen Opiskelija Töihin -sivustolle, jossa etsittiin tekijää toteuttamaan sisällönhallintajärjestelmä päivityksenä toimeksiantajan asiakkaalle. Otettuani toimeksiantajaan yhteyttä, sain onnekseni työn alle kyseisen projektin opinnäytetyönä. Kiinnostuin projektista erityisesti siksi, koska siihen liittyi sellaisia tekniikoita, joihin en ollut vielä tutustunut. Lisäksi osaamisen kehittäminen ja syventäminen jo hieman tutuiksi tulleiden tekniikoiden osalta kiinnosti.

Opinnäytetyön aikana suunniteltiin ja kehitettiin verkkosivustolle sisällönhallintajärjestelmä hyödyntäen Laravel 8-ohjelmistokehystä. Opinnäytetyön kohteena oleva sivusto oli Single Page Application-tyyppinen ja oli Shade Empire -yhtyeen virallinen sivusto. Sivustolla ylläpidettäviä asioita ovat esimerkiksi keikat, yhtyeen jäsenet ja erilaiset linkit sekä taustakuvat. Projektin aikana toteutettiin sopiva tietokanta, käyttöliittymä sekä niiden välinen kommunikaatio. Lopputuloksena toimiva sisällönhallintajärjestelmä sivustolle.

1.1 Toimeksiantaja

Traakkoni Oy on kuopiolainen vuonna 2019 perustettu ohjelmistoalan yritys. Traakkoni Oy:n erikoisalaa ovat web-pohjaiset ratkaisut, kuten Single Page Applicationit (SPA) sekä erilaiset suuremmat sivustokokonaisuudet. Toimeksiantajalla on myös kokemusta PHP Frameworkista Laravelista ja toimeksiantajan pyynnön mukaisesti opinnäytetyö rakennettiin Laravelia käyttäen.

1.2 Lyhenteet ja määritelmät

Backend	Palvelinpuolen toteutukset
Frontend	Käyttöliittymän toteutus
PHP	Hypertext Preprocessor. Ohjelmointikieli, back-end toteutuksiin
Laravel	PHP ohjelmistokehys.
CMS	Content Management System, sisällönhallintajärjestelmä
MVC-malli	Ohjelmistoarkkitehtuurimalli, jossa käytetään malleja (Model), näkymiä (View) sekä ohjaimia (Controller)
WAMP	Kokonaisuus, joilla voidaan toteuttaa web-palvelujen kokonaisuus. Windows, Apache, MySQL, PHP.
PHPStorm	Jetstreamin kehittämä ohjelmointiympäristö.
Composer	Laravel-projektien pakettimanageri. Apuna muun muassa Laravelia asennettaessa
Artisan	Komentorivipohjainen apuri Laravel-projekteissa.
WYSIWYG	What You See Is What You Get, ohjelma, joka näyttää lopputuloksen jo muokkausvaiheessa.
Validointi	Tiedon oikeellisuusuuden määrittely.
Javascript	Ohjelmointikieli websovelluksissa.
CSS	Cascading Style Sheets. Ulkoasun määrittely HTML:ssä.
Bootstrap	CSS sovelluskehys responsiivisten web-sovellusten toteuttamiseen.
Tailwind.css	CSS sovelluskehys, kuten Bootstrap.
Fortify	Laravel Jetstreamin käyttämät backend-komponentit autentikointiin.
Jetstream	Laravel scaffold, joka luo komponentit kirjautumiselle ja rekisteröitymiselle.
Livewire	Laravel kirjasto käyttöliittymien luontiin
SQL	Structured Query Language. Kyselykieli, jonka avulla voidaan esimerkiksi hakea tai poistaa dataa relaatiomallisesta tietokannasta.
Git	Versionhallintatyökalu lähdekoodille.
Route	Määrittely, mikä sivu vastaa kutsuttassa jotakin URL:ia.
CRUD	Create, Read, Update, Delete. Operaatiot, joiden avulla voidaan luoda, lukea, päivittää ja poistaa dataa.
Migraatio	Laravelin versionhallinta tietokannoille.
CSRF	Cross-site Request Forgery, tietoturvahyökkäys HTTP-kutsun avulla.
Autentikointi	Käyttäjän todentaminen.
Blade	Laravel MVC-mallin View-komponentti.

2 SISÄLLÖNHALLINTAJÄRJESTELMÄ

Content Management System (CMS) eli sisällönhallintajärjestelmä tarkoittaa järjestelmää, jonka kautta käyttäjät voivat hallita erilaisia sisältöjä esimerkiksi verkkosivustolla. Sisällönhallintajärjestelmän tarkoitus on tehdä kohteestaan dynaaminen kokonaisuus, jossa tietoa voidaan lisätä, päivittää, tarkastella sekä poistaa. Sisällönhallintajärjestelmän etuja on se, että peruskäyttäjä ei tarvita laajaa tietoteknistä osaamista sivuston ylläpitämiseen. (Ite Wiki)

Sisällönhallintajärjestelmiä voidaan luoda itse tai käyttää valmiiksi rakennettuja paketteja. Sisällönhallintajärjestelmää luodessa on tärkeää suunnitella, mistä osista se koostuu ja millä tekniikoilla toteutus tapahtuu. Järjestelmä voi koostua web-pohjaisista visuaalisista komponenteista sisällön sekä hallintapaneelin osalta. Näiden lisäksi tarvitaan taustalle logiikka tiedonkulkuun sekä tietokanta, johon sisällöt saadaan tallennettua. Sisällönhallintajärjestelmän voi rakentaa useilla eri ohjelmistokehyksillä ja kielillä. Web-pohjaisissa sisällönhallintajärjestelmissä käyttöliittymä (frontend) koostuu esimerkiksi HTML:stä, CSS:stä ja Javascriptistä ja palvelinpuolen toteutus esimerkiksi PHP:stä.

On myös olemassa valmiita sisällönhallintajärjestelmiä. Valittaessa sisällönhallintajärjestelmää, tulisi selvittää miten sivuston on tarkoitus toimia, millaisia ja miten laajoja kokonaisuuksia siihen tarvitaan. Järjestelmiä verrattaessa kannattaa myös tiedostaa, tarvitaanko sivuston rakentamiseen erityisosaamista sekä onko apua saatavilla esimerkiksi asiakastuesta tarpeen tullen. Selvitetään, löytyykö tarvittavat komponentit valmiina vai joudutaanko jotain rakentamaan itse käsin. Suurimmissa sisällönhallintajärjestelmissä on valtavat määrät erilaisia liitännäisiä, teemoja ja muita toiminnallisuuksia, mutta joissakin on tiettyihin tilanteisiin paremmat ominaisuudet. Näiden asioiden lisäksi, kannattaa tutustua sisällönhallintajärjestelmien hinnoitteluvaihtoehtoihin. (Medium)

Wordpress on maailman käytetyimpiä sisällönhallintajärjestelmiä ja se pohjautuu avoimeen lähdekoodiin ja se on erittäin skaalautuva eri käyttötarkoituksissa. Käyttötarkoituksia voi olla esimerkiksi verkkokauppa, yrityksen kotisivu tai vaikkapa blogisivusto. Wordpressin hinta vaihtelee paljon sen mukaan, millaisia palveluita, liitännäisiä ja ylläpitoa siihen halutaan. Wordpressistä on saatavilla myös ilmainen versio. (Automattic)

Vaihtoehtoja on Wordpressin lisäksi lukuisia, kuten suomalaisten kehittämä Joomla. Useat niin sanotusti valmiit sisällönhallintajärjestelmät pohjautuvat PHP:hen. Wordpressin ja Joomlan lisäksi tähän kategoriaan kuuluvat myös Drupal ja tämä kolmikko kuuluu tämän hetken suosituimpiin sisällönhallintajärjestelmiin. Sisällönhallintajärjestelmiä on kehitetty myös muilla tekniikoilla ja ohjelmistokehyksillä. Tällaisia ovat esimerkiksi .NET-ohjelmistokehyksen variaatiot sekä Javaan pohjautuvat järjestelmät. (Ionos)

Tämän opinnäytetyön aiheena oli rakentaa sisällönhallintajärjestelmä ja se toteutettiin Laravel Frameworkilla ja sen komponenteilla.

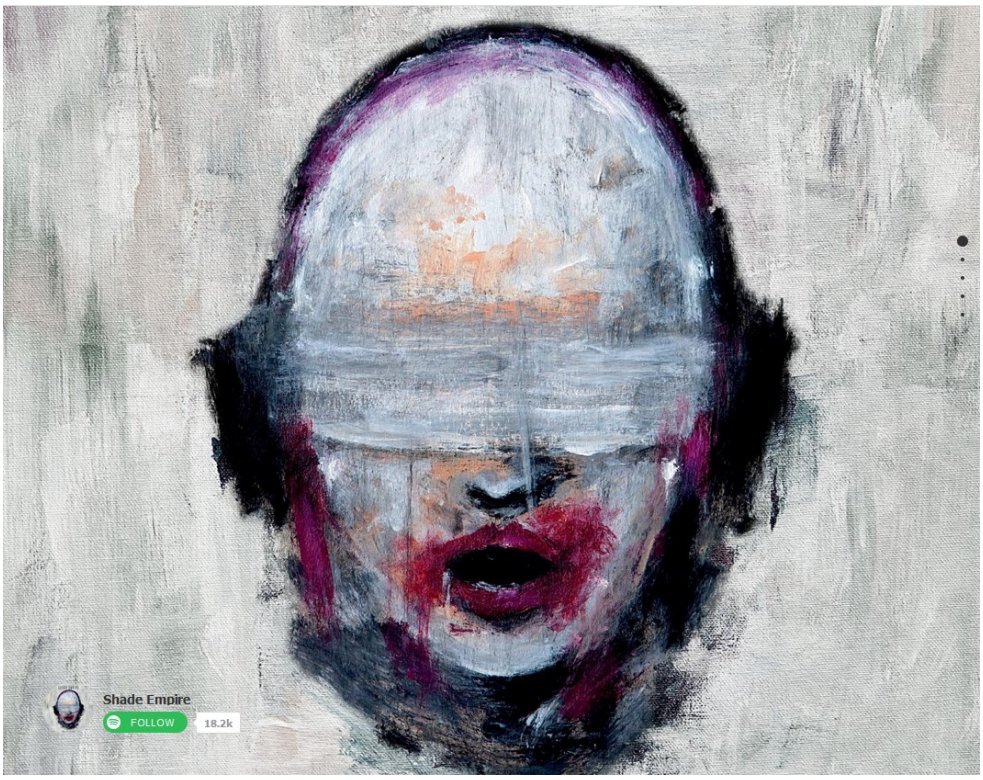
3 TAUSTATIEDOT

3.1 Lähtötilanne

Tässä opinnäytetyössä rakennettiin sivuston ylläpitojärjestelmä Single Page Application-sivustolle (SPA). Single Page Application on verkkosivusto, jonka sisältö on usean sivun sijaan yhdellä sivulla. Sivuston sisältöä käydään läpi käyttämällä hiiren rullaa, paljastaen uutta sisältöä. Joissain toteutuksissa on mahdollista myös navigoiminen sivun osien välillä, varsinkin jos osiota on paljon. Tällä sivulla navigaatiota osien välillä ei ole. SPA:n toteutuksessa käytetään tyypillisesti HTML:n ja CSS:n lisäksi Javascriptiä sekä Bootstrapia.

Opinnäytetyön kohteena oleva sivusto koostui Single Page Applicationin viidestä osiosta, joissa kahdessa oli sisältönä sivusuunnassa siirrettävää sisältöä. Verkkosivuston aloitustilanne oli se, että kyseinen HTML-sivu sisälsi ainoastaan kovakoodattua dataa. Tämä tarkoitti sitä, että kaikki muutokset oli tehtävä suoraan koodiin ja sivuston omistajilla ei ollut osaamista tai mahdollisuutta päivittää sitä itse. Tilanne oli se, että muutokset tehtiin sivustolle ottamalla yhteyttä sivuston ylläpitäjään, jolle ilmoitettiin halutut muutokset ja sivuston ylläpitäjä toteutti ne.

Sivuston rakenne koostuu useasta eri näkymästä, joissa kussakin oli erityyppisiä sisältöjä. Sisältöjä olivat erilaiset tekstit, otsikot, kuvat, linkit ja listat.



Kuva 1 – Muokattavan sivuston aloituskuvana yhtyeen albumin kansi

3.2 Opinnäytetyön tavoitteet ja aiheen rajaus

Opinnäytetyön ensisijainen tavoite oli muuttaa staattinen sivusto dynaamiseksi. Tätä varten tuli suunnitella ja toteuttaa käyttöliittymä sisällönhallintajärjestelmälle. Käyttöliittymän tuli sisältää ylläpi-

täjälle näkymä, jossa datan muokkaukseen käytettäisiin WYSIWYG-editoria tai HTML form-elementtiä. Jokaiselle osiolle luotaisiin sivu, johon tuli luoda tarvittavat CRUD-operaatiot. CRUD-operaatiot (Create Read Update Delete) ovat uuden datan luominen, olemassa olevan datan tarkastelu ja päivittäminen sekä poistaminen.

Käyttöliittymän lisäksi suunniteltiin tietokanta sekä kannan ja ohjelman välinen logiikka sekä tarvittavat toiminnot. Käytettäväksi tietokannaksi oli valikoitunut MySQL ja siihen piti pystyä tallentamaan ainakin käyttäjien kirjautumistiedot sekä sisällönhallintajärjestelmän käyttämä data. Sisällönhallintajärjestelmässä käytettävä data oli suurimmaksi osaksi tekstiä sekä kuvia.

Sivuston ylläpitäjille tuli toteuttaa sivustoa varten autentikointi, jotta vain halutut henkilöt pääsevät järjestelmään. Tämä tarkoitti myös uusien käyttäjien lisäystä sekä vanhojen poistoa vain kirjautuneille käyttäjille. Yksi vaihtoehto toteutukseen heti alkuvaiheessa oli autentikoinnin ja rekisteröinnin toteutukseen oli Laravel Jetstream sekä Laravel Livewire-kirjaston yhdistelmän käyttö.

Opinnäytetyö rajattiin sisällönhallintajärjestelmän toteutukseen ja itse julkinen sivusto jätettäisiin ennalleen, lukuun ottamatta koodia, joka luo sivulle dynaamisuuden. Opinnäytetyön toteutussuunnitelma purettiin pienempiin osiin työn suunnitteluvaiheessa.

Opinnäytetyön tavoitteena oli myös tutustua Laravel-ohjelmistokehykseen ja varsinkin sen MVC-malliin sekä syventää osaamista muiden tekniikoiden, kuten Javascriptin ja Git-versionhallinnan osalta. Opinnäytetyön osana oli tarkoitus lisäksi dokumentoida ohjelmointiympäristön ja Laravelin asennusta ja käyttöä Windows-ympäristössä.

3.3 Työn suunnittelu

Opinnäytetyön aihe oli rajattu olemassa olevan verkkosivun CMS:n (Content Management System) eli sisällönhallintajärjestelmän toteutukseen. Käyttöön ei haluttu ottaa raskasta järjestelmää, koska itse hallittava sivusto ei ollut laaja eikä siihen tarvittu ottaa esimerkiksi Wordpressin tarjoamia lisäosia käyttöön. Tämän opinnäytetyön aikana ei myöskään ollut tarkoitus koskea hallittavan sivuston lähdekoodiin muilta osin kuin tuoda data tietokannasta staattisen koodin sijaan.

Opinnäytetyön suunnitteluvaihe alkoi opinnäytetyön aiheen valinnan jälkeen. Ensimmäinen asia oli valita työssä käytettävät tekniikat ja työkalut sekä miettiä, kuinka käyttäjäystävällisyys tulisi huomioida projektin edetessä. Tämä vaati taustatyön tekemisen käyttöön tulevista tekniikoista ja työkaluista ja tiedonhakuun tuli jättää tarpeeksi aikaa. Suunnitteluvaiheessa tiedonhakuun käytettiinkin suurin osa ajasta.

Toimeksiantajalta saatujen ehdotusten perusteella päädyttiin käyttämään työssä Laravel ohjelmistokehystä. Laravel 8-versio oli julkaistu hiljattain ja siitä oli julkaistu myös jo stabiili versio, joten tässä työssä käytettiin Laravel 8-versiota. Muutoksia Laravel 8-versiossa aiempiin verrattuna oli tapahtunut jonkun verran. Koska aiempaa kokemusta Laravelista ei ollut, ei näillä muutoksilla ollut juurikaan väliä. Tiedonhaussa tuli kuitenkin huomioida, että tarkasteltavat ohjeet koskivat oikeaa versiota, mikäli kyseiseen komponenttiin oli tullut uudessa versiossa muutoksia. Laravel tarjoaa erittäin hyvän ja selkeän dokumentaation ohjelmistokehittäjille, jota oli helppo seurata.

Koodieditoriksi valittiin PHPStorm toimeksiantajalta saatujen suositusten pohjalta. Koodieditorina ei olisi tarvinnut käyttää juuri PHPStormia, vaan tähän olisi käynyt esimerkiksi myös Visual Studio Code tai muu vastaava ohjelmisto. Pelkkää tekstieditoria ei kuitenkaan voi suositella käytettäväksi tällaisessa projektissa. PHPStormin kaltaiset ohjelmat tarjoavat parempia toiminnallisuuksia esimerkiksi projektinhallintaan. Ohjelman rakenne on helppo pitää siistissä järjestyksessä ja tällaisen ohjelman kautta rakenne on helpompi ymmärtää. Näiden lisäksi myös esimerkiksi tietokantayhteys ja salattu palvelinyhteys ohjelman kautta suoraan on mahdollista.

Käyttäjäystävällisyyttä pyrittiin lisäämään etenkin tekemällä sivustosta responsiivinen. Ohjelmistoa tuli voida käyttää tietokoneella sekä myös mobiililaitteilla. Tätä varten todettiin, että Bootstrap olisi hyvä ottaa mukaan toteutukseen.

Suunnitteluvaiheessa piti myös valita paikallinen ympäristö, jossa ohjelmaa voitaisiin ajaa. Saatavilla oleva käyttöjärjestelmä oli Windows, johon asennettiin WampServer. WampServer tuntui hyvältä vaihtoehdolta, koska sen käytöstä oli jo aiempaa kokemusta. WampServer asennuspakettiin kuuluu Apache, MySQL sekä PHP.

Ennen itse työn aloittamista projekti paloiteltiin pienempiin osiin ja niille annettiin karkeahkot rajaukset halutuista toiminnoista sekä aikataulut, jossa osan tulisi olla valmis. Tarkoituksena oli käyttää ketterän kehityksen menetelmiä. Ketterään kehitykseen kuuluu nopea reagointi tarvittavien muutosten tekemiseen. Projektin edetessä tämä tarkoitti sitä, että mikäli huomattiin jostakin toiminnosta puuttuvat jotain, se lisättiin listalle ja toteutettiin. Vaihtoehtona tässä olisi pitäytyä tiukasti suunnitelmassa, mutta tällaisen projektin kohdalla ketterä menetelmän valitseminen tuntui luontevalta.

4 KÄYTETYT TYÖKALUT JA TEKNIIKAT

4.1 Työkalut

Tässä osiossa käydään läpi tärkeimmät tämän opinnäytetyön aikana käytetyt työkalut. Työkalujen valinta tapahtui toimeksiantajan ohjeistuksista ja suosituksista.

4.1.1 WampServer

WampServer on kokoelma ohjelmistoja, joilla saadaan toteutettua toimiva ympäristö web-sovellusten paikalliseen kehittämiseen, tai myös verkkosivuston julkaisemiseen. WampServer on Windows-ympäristöön tarkoitettu paketti. Pakettiin kuuluu Apache, MySQL sekä PHP. Apache on HTTP-palvelin, joka pohjautuu avoimeen lähdekoodiin. MySQL on relaatiomalliin pohjautuva tietokanta ja sen hallinta tapahtuu paketin mukana tulevan käyttöliittymän, PhpMySQL:n kautta. (Bourdon) PHP on käytettävä ohjelmointikieli, jolla itse sovellus rakennetaan. PHP:stä on paketissa valittavana useita versioita, joista kannattaa ottaa käyttöön tarpeeksi tuore versio ja tarkistaa, että kyseiselle versiolle löytyy edelleen tuki (The PHP Group).

WampServerin käytölle on vaihtoehtoja. Vastaavanlaiset ohjelmistopaketit on saatavilla myös muille käyttöjärjestelmille. LAMP on Linux-ympäristöön tarkoitettu paketti ja MAMP:ia voidaan käyttää Mac-kehityksessä. XAMPP on niin kutsuttu cross-platform paketti ja sitä voidaan ajaa Windows, Linux tai Mac-ympäristöistä. (Code Boxx)

4.1.2 PhpStorm

PhpStorm on Jetstreamin kehittämä ohjelmointiympäristö. Ympäristö sisältää ohjelmistoprojektille monia hyödyllisiä ominaisuuksia. Usempien ohjelmointiympäristöjen tapaan, myös PhpStorm etuja ovat projektinhallinnalliset toiminnot. PhpStormia suositellaan käytettäväksi esimerkiksi Symfony ja Laravel -ohjelmistokehysten kanssa sekä sitä voi myös käyttää kehittäessä esimerkiksi Wordpressiä tai Joomlaa. PhpStormin hyviä puolia ovat muun muassa komentorivi komentojen ajamiseen, salatut yhteydet palvelimeen tai ulkoiseen tietokantaan sekä versionhallintaintegraatio, jonka avulla pysytään helposti ajan tasalla tehdyistä muutoksista. (Jetbrains)

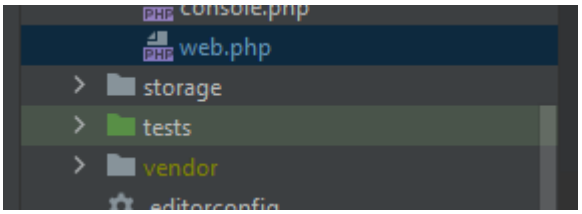
4.1.3 Git

Git on versionhallintatyökalu, jonka tarkoitus on pitää lähdekoodi hallinnassa riippumatta siitä, kuinka monta kehittäjää projektilla on. Projektin tai kehitystyön alkuvaiheessa kannattaa luoda oma repositorio eli paikka GitHubin kautta, johon lähdekoodi otetaan talteen. GitHubin käyttöön tulee luoda tunnukset. Kun on luonut repositorion GitHubiin, siihen voidaan viitata paikallisesta ympäristöstä. Määrittelyt tehdään projektikansion juuressa ja ohjeistus komentoihin löytyy GitHubista, kun repositorio on luotu (GitHub, Inc, 2020). Ajettavat komennot ohjeistuksen mukaan ovat:

```
echo ”# oppari” >> README.md //README.md tiedoston lisäys
git config –global user.email [käyttäjän e-mail] //määritellään sähköposti
git config –global user.name [“käyttäjän nimi”] //nimi
git init //luo tyhjän repositorion
git commit -m “first commit” //
```

```
git branch -M main //
git remote add origin https://github.com/projektin_polku //yhdistetään
git push -u origin main //viedään paikallinen sisältö GitHubiin
```

Kun projektille on luotu paikallinen repositorio ja se on yhdistetty GitHubiin, voidaan aina halutessa viedä tehdyt muutokset viedä myös versionhallintaan. PHPStormissa on ominaisuus, jonka avulla voidaan nähdä, mikäli tiedostoa on muutettu viimeisen repositorioon tallennuksen jälkeen. Tiedosto näkyy tällöin vaaleansinisenä. Tiedostot, joita ei viedä versionhallintaan näkyvät tummanvihreinä.



Kuva 2 - PHPStorm versionhallinnan värikoodaus

Projektissa tapahtuneet muutokset viimeisimmän Git:iin viennin jälkeen saa myös selville ajamalla komento:

```
git status
```

Projektin muutokset viedään komennoilla

```
git add .
git commit -m "saatetekstı"
git push origin master
```

Mikäli kaikki halutaan tuoda muutokset GitHubista, voidaan ajaa komento:

```
git pull origin main
```

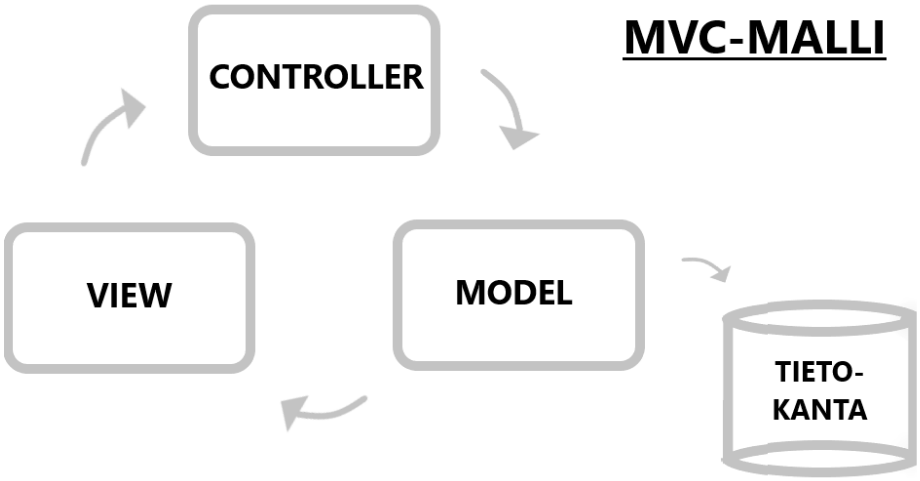
4.2 Laravel

Laravel on PHP:n ohjelmistokehys, jonka avulla on helppo luoda toimivia ja hyvännäköisiä sovelluksia. Laravel on melko yleinen web-ohjelmistokehys kehittäjien parista. Statistan mukaan vuoden 2020 alussa Laravel käytti noin 11% prosenttia kehittäjistä (Statista Inc.). Laravelin viimeisin vakaa versio on 8 ja se on julkaistu 8. syyskuuta 2020. Versio 6 ja 7 on edelleen tuettuja vuonna 2020. Versiolle 6 on luvattu tietoturvatuki 2022 ja bugikorjaukset 2021 lokakuuta saakka. Uutta Laravelin versiossa 8 on esimerkiksi Laravel Jetstream. Laravel Jetstream tarjoaa komponentit muun muassa kirjautumista sekä rekisteröintiä varten valmiina pakettina. Jetstream (Taylor Otwell)

Laravel on monikäyttöinen ohjelmistokehys, jonka hyvinä puolina voidaan mainita muun muassa tuki MVC-mallille, turvallisuus, suuri valikoima kirjastoja ja komponentteja sekä migraatiot. Tässä kappaleessa kuvataan Laravel ohjelmistokehyksen perusominaisuuksia.

4.2.1 MVC

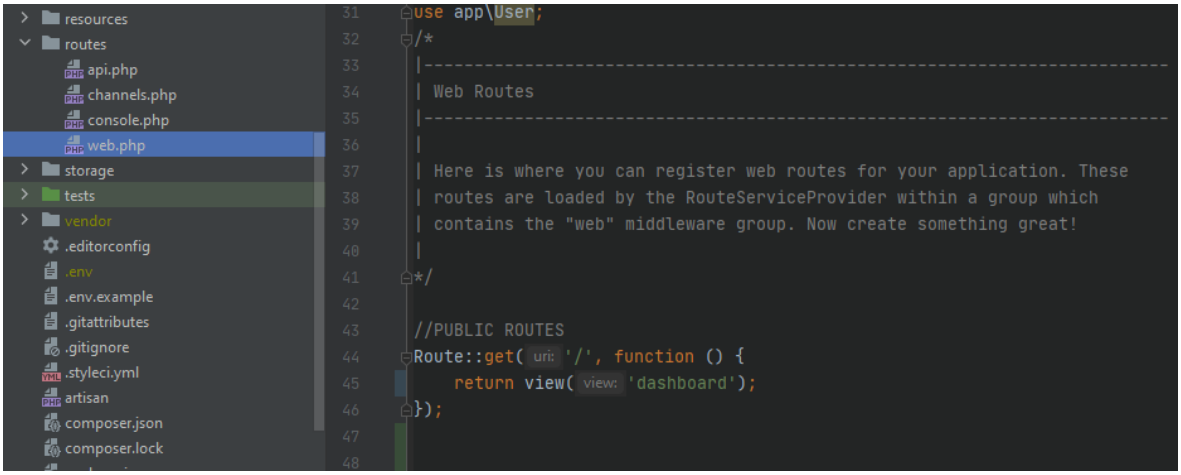
Laravel noudattaa MVC-arkkitehtuurimallia. MVC-malli koostuu kolmesta komponentista, mallista (Model), näkymästä (View) sekä ohjaimesta (Controller). Näkymä on esimerkiksi jonkinlainen käyttöliittymä, joka lähettää pyynnön ohjaimelle esimerkiksi poistaa dataa. Ohjain lähettää tiedon edelleen mallille, joka on yhteydessä tietokantaan. Näkymätiedostoissa on Laravel ohjelmistokehyksessä blade.php-tiedostopääte.



Kuva 3 - Model View Controller -malli

4.2.2 Routing

Laravel ohjelmistokehyksessä kaikki polut (route) määritellään Route-kansiossa olevissa tiedostoissa. Kansiossa sijaitsevat tiedostot ladataan automaattisesti ohjelmistokehyksen ansiosta. Tiedostossa web.php olevat tiedostot liittyvät web-rajapinnan reitteihin. Web-reiteille on oma middleware-ohjelmisto, joka hoitaa esimerkiksi CSRF-suojauksen.



Kuva 4 - Reititysten määrittely web.php -tiedostoon

CSRF on verkkohyökkäys, jonka avulla hyökkääjä voi yrittää syöttää haittakoodia HTTP-pyyntöjen avulla (Taylor Otwell). Jotta CSRF-suojaus toimisi, on määriteltävä tarvittaviin kohtiin @CSRF-merkintä. Tällaisia kohtia ovat esimerkiksi Form-elementit, jossa käyttäjä voi syöttää dataa verkon yli.

Reitteihin voidaan lisätä middleware-funktioita ja niitä voidaan myös luoda itse. Artisan-komento

```
php artisan make:middleware MiddlewareName
```

luo uuden Middleware-komponentin, jota voidaan muokata halutunlaiseksi.

4.2.3 Migraatiot

Migraatio on Laravel ohjelmistokehyksessä versionhallinta tietokannalle. Käytännössä tämä tarkoittaa sitä, että joka kerta kun tietokantaan halutaan tehdä rakenteellinen muutos, se tehdään migraation kautta. Tällainen muutos voi olla esimerkiksi uuden taulun luominen, taulun uudelleennimeäminen tai taulun datatyypin muutos. Artisan-komennolla

```
php artisan make:migration create_taulunnimi_table
```

saadaan tehtyä uusia migraation. Artisan luo valmiin migraatio-tiedoston ja esimerkiksi uutta tietokantataulua luodessa up-funtioon määritellään tietokantataulun malli (Schema) create-funktiossa. Tietokantataululle annetaan nimi ja toisena parametrina Blueprint-objekti, jonka avulla voidaan määritellä rakenne taululle.

```php
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'users', function (Blueprint $table) {
            $table->id();
            $table->string( column: 'name');
            $table->string( column: 'email')->unique();
            $table->timestamp( column: 'email_verified_at')->nullable();
            $table->string( column: 'password');
            $table->rememberToken();
            $table->foreignId( column: 'current_team_id')->nullable();
            $table->text( column: 'profile_photo_path')->nullable();
            $table->timestamps();
        });
    }
```

Kuva 5 - Tietokantaskeeman luonti migraatiossa

Uuden sarakkeen luominen olemassa olevaan tietokantatauluun onnistuu luomalla uusi migraatio, jonka up-funktioon luodaan skeeman table-funktio. Table-funktioon määritellään lisättävä sarake samalla tavoin, kuin uuden taulun luomisessa.

```php
public function up()
{
    Schema::table( table: 'tablename', function (Blueprint $table) {
        $table->string( column: 'newcolumn');
    });
}
```

Kuva 6 - Sarakkeen lisääminen tauluun

Tietokantataulun sarakkeen muokkaamista varten projektiin tulee lisätä Doctrine DBAL -kirjasto. Tämä tapahtuu ajamalla Composer-komento:

```
composer require doctrine/dbal
```

Tämän jälkeen voidaan luoda uusi migraatio, jolloin table-funktioon voidaan määritellä haluttu muutos. Sarakkeen määrittelyyn tulee lisäksi lisätä perään ”->change()”-funktion, jotta muutos tehdään.

```php
public function up()
{
    Schema::table( table: 'tablename', function (Blueprint $table) {
        $table->text( column: 'columnname')->change();
    });
}
```

Kuva 7 - Sarakkeen muokkaus

Migraatiot ajetaan Artisan-komennolla, jolloin migraatiossa määritellyt muutokset tehdään.

```
php artisan migrate
```

Mikäli tehty migraatio halutaan perua, voidaan ajaa komento, joka peruu viimeksi ajetun migraatiokomennon:

```
php artisan migrate:rollback
```

Rollback-komentoon voidaan myös määritellä, kuinka monta migraatiota halutaan perua antamalla komennon perään option, jossa x on peruttavien migraatioiden määrä.

```
php artisan migrate:rollback --step=x
```

4.2.4 Tietoturva

PHP:tä kielenä on kritisoitu siitä, ettei se ole pysynyt ajan tasalla muiden web-tekniikoiden kanssa. PHP:hen on mahdollista rakentaa käsin erilaisia komponentteja, jotka nostavat tietoturvaa. Tämä saatetaan kuitenkin kokea hankalaksi, koska muissa tekniikoissa nämä ovat huomioitu paremmin esimerkiksi valmiiden funktioiden avulla. Tilanne voi myös olla se, että kehittäjä ei välttämä näe tarpeelliseksi päivittää sivua käyttämään uudempaa versiota ja tietoturva-aukkoja syntyy. (Light IT)

Laravel on luotu helpottamaan ohjelmistokehittäjien työtä laajalla kirjolla valmiita komponentteja ja valinnaisia toiminnallisuuksia. Laravel on tämän lisäksi ajateltu paikkaavan PHP:n ongelmia esimerkiksi tietoturvan osalta.

4.2.5 Blade

Opinnäytetyön aikana luotiin useita view- eli näkymäkomponentteja. Näkymäkomponenteilla on Laravel-maailmassa blade.php-pääte. Blade-komponentit ajetaan Blade moottorin läpi, jonka hyvänä puolena on se, että koodi voi sisältää sekä pelkkää PHP-koodia tai HTML:ää, sekä niiden sekoitusta (Taylor Otwell). Blade-dokumentissa voidaan sijoittaa ehtolauseita lisäämällä @-merkki ehtolauseen eteen. Controller-tiedoston avulla voidaan palauttaa dataa näkymätiedoston käyttöön. Data saadaan näkyviin esimerkiksi käymällä se @foreach-rakenteessa läpi, mikäli kyseessä on taulukko.

```
@foreach($users as $user)
    <tr>
        <td>{{$user->name}}</td>
        <td>{{$user->email}}</td>
        <td>{{$user->created_at}}</td>
    </tr>
@endforeach
```

Kuva 8 - Foreach loop Laravelissa

Blade-tiedostoissa voidaan käyttää HTML:ää ja sen elementtejä. Form-elementtejä käyttäessä tulee muistaa lisätä @CSRF-merkintä elementtiin tietoturvan lisäämiseksi.

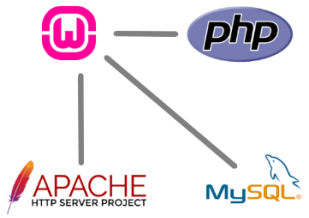
5 TYÖN TOTEUTUS

5.1 Ympäristön asennus

Opinnäytetyö kehitys toteutettiin paikallisesti Windows-ympäristössä. Opinnäytetyön kehitysympäristön asennus aloitettiin tarkistamalla Laravelin vähimmäisvaatimukset työasemalle, johon paikallinen ympäristö luotaisiin. PHP minimiversio Laravel 8 asennukseen on 7.3. ja seuraavat PHP:n laajennukset tuli olla asennettuna:

- BCMath
- Ctype
- Fileinfo
- JSON
- Mbstring
- OpenSSL
- PDO
- Tokenizer
- XML

Laravelia käyttäessä voisi myös asentaa Laravel Homesteadin. Laravel Homestead on virtuaaliympäristö, jossa vaatimusten toteutus on esimääritelty. Tämä tarkoittaa sitä, että PHP sekä web-palvelin on jo käytännössä asennettu aloittaessa projektin rakentamisen. Tämän opinnäytetyön aikana päädyttiin kuitenkin käyttämään WampServeriä, koska se oli ennestään tuttu. WampServerin asennuspakettiin kuuluu PHP:n lisäksi MySQL sekä Apache.



Kuva 9 - WampServer: Apache, PHP, MySQL

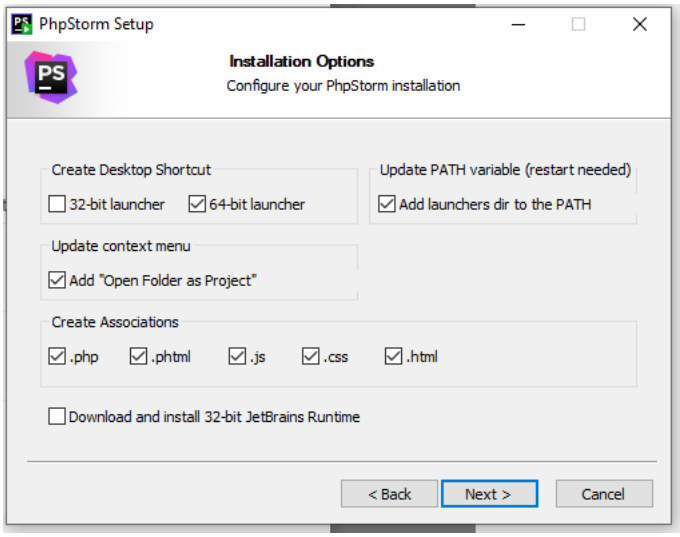
PHP asennettiin WampServer-ohjelmistopaketin asennuksen yhteydessä ja käytetty PHP:n versio oli 7.3.21. Kaikki muut alle vaaditun versionumeron kannattaa jättää pois asennuksesta, koska niitä ei tarvita ja ne saattavat sotkea konfiguraatioita. Opinnäytetyön aikana esiintyi tähän liittyvä ongelma, koska WampServer paketin asennuksen yhteydessä PHP:stä oli lipsahtanut mukaan sekä uudempi versio 7.3, mutta myös vanhempi versio 5.6. Tämä aiheutti sen, ettei esimerkiksi jotain pakettia voitu asentaa, koska Laravel-projekti meni versioiden kanssa sekaisin. Asia ratkaistiin poistamalla PHP:n vanha versio kokonaan ja luomalla uusi Laravel-projekti. Tämä oli mahdollista, koska työ oli vasta alkutekijöissään. Mikäli projekti olisi ollut pidemmällä, konfiguraatioita olisi todennäköisesti jouduttu kaivamaan syvemmältä. Ennen Laravelin asennusta tulee myös tarkistaa tarvittavat PHP laajennukset. Jos jokin laajennuksista puuttuu, ne voi valita käyttöön WampServerin PHP-valikosta kohdasta ”PHP extensions”.

Paikallisen ympäristön lisäksi, opinnäytetyötä varten pystytettiin toimeksiantajan puolesta myös palvelin, joka mukailisi tuotantoympäristöä. Tämä palvelin oli käytännössä työn testausta varten, ennen työn siirtoa oikealle tuotantopalvelimelle. Testauspalvelimella havaittiin, että sieltä puuttui PHP:n PDO-laajennus ja se täytyi ottaa käyttöön, jotta sovellus saatiin toimimaan.

Paikallisena tietokantana käytettiin WampServerin mukana tullutta MySQL:ää ja sen hallinta tapahtui PHPMyAdminin kautta. Kun WampServer oli asennettu, aloitettiin itse Laravelin asennus.

Kuva 10 - PhpMyAdmin kirjautuminen

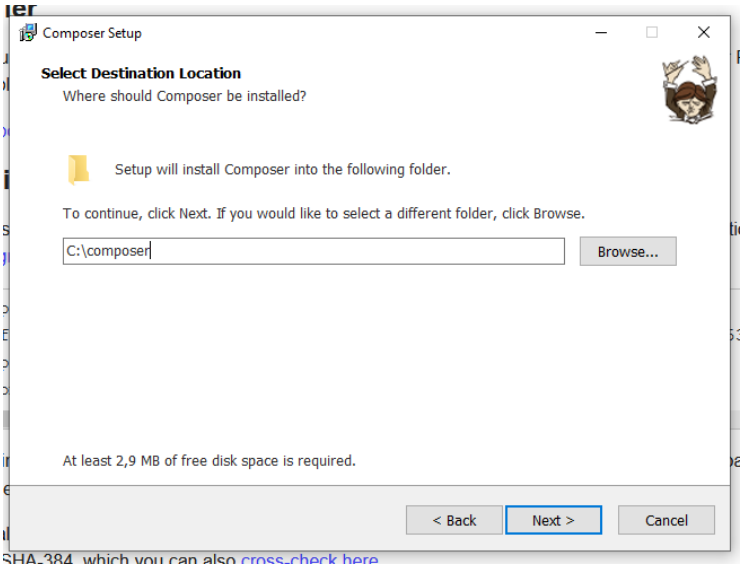
Editorina opinnäytetyön aikana käytettiin PhpStormia. Asennuspaketin saa ladattua JetBrainsin PhpStorm-sivuilta. PhpStormista saa 30-päivän kokeiluversion ja opiskelijat voivat saada vuoden lisenssin ilmaiseksi.



Kuva 11 - PhpStorm asennus

Laravelin asentamiseen vaaditaan Composer. Composer on PHP-projektin pakettienhallintaan keskittyvä ohjelmisto, mutta se ei ole suoranaisesti paketinhallintaohjelmisto. Erona normaaleihin paketinhallintaohjelmiin, Composer ei asenna paketteja tai kirjastoja globaalisti, vaan projektinsisäisesti (Composer). Myös Composer vaatii PHP:n, joten esimerkiksi WampServerin asennus tulee tehdä en-

nen Composerin asennusta, mikäli WampServeriä käytetään. Composerin asennusmedian saa ladattua sen verkkosivulta. Tärkeä huomio asennuksessa on määritellä PATH-ympäristömuuttuja, jotta Composeria voidaan ajaa eri sijainneista.



Kuva 12 - Composer asennus

Composerin asennuksen jälkeen tulee ladata Laravelin asennuspaketti. Tähän käytetään juuri asennuttua Composeria ajamalla tarvittava komento:

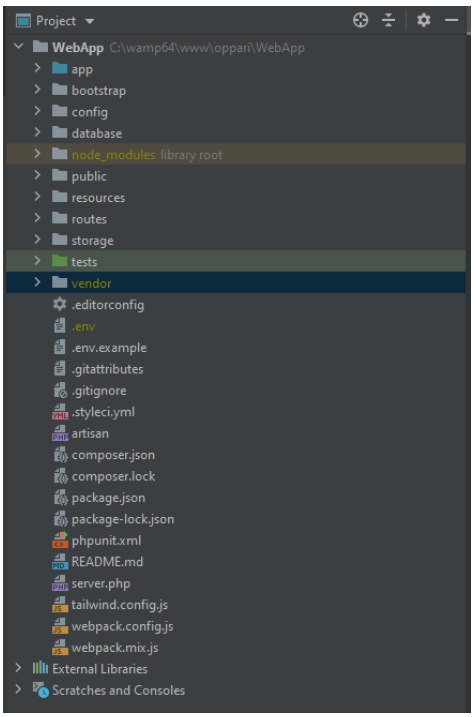
```
composer global require laravel/installer
```

Asennuksen jälkeen voidaan ajaa komento halutussa tiedostosijainnissa, joka luo uuden Laravel-projektin tarvittavine riippuvuuksineen:

```
laravel new [ohjelmannimi]
```

Kun projekti on luotu, kaikki siihen liittyvät komennot ajetaan projektikansiossa. Toinen tapa luoda Laravel-projekti on Composerin avulla, jolla tavoin tämän opinnäytetyön aikana tehtiin. Se toteutetaan ajamalla komento:

```
composer create-project –prefer-dist laravel/laravel [projektinnimi]
```



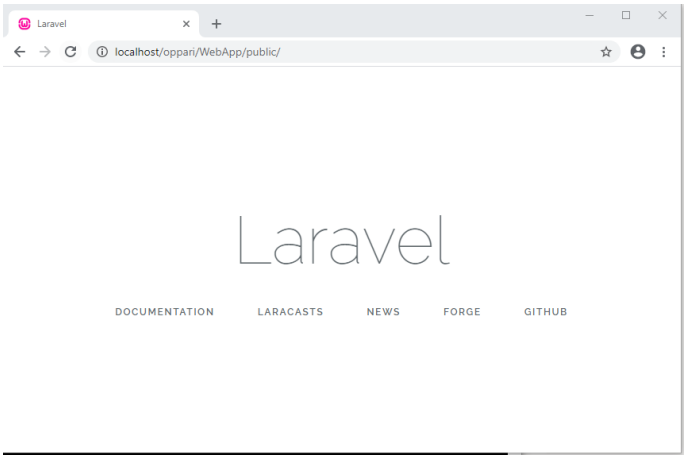
Kuva 13 - Laravel projektin rakenne

Laravel-projektin asennuksen yhteydessä env-tiedostoon luodaan Application Key, jota käytetään istunnoissa, salauksessa sekä CSRF-tokenin luomisessa. Mikäli env-tiedostosta puuttuu 32-merkkiä pitkä Application Key, se voidaan luoda ajamalla komento

```
php artisan key:generate
```

Tämä tiedosto on muutoinkin tärkeä, siellä on määritelty ympäristön konfiguraatiot, kuten esimerkiksi tietokannan määrittelyt sekä debug-tila virheenmääritystä varten. Env-tiedosto sisältää myös paikan tietokannan määrittelylle. Tiedosto on ympäristökohtainen, joten projektia vietäessä esimerkiksi tuotantopalvelimelle, tätä tiedostoa ei siirretä muun projektin mukana versionhallinnassa.

Kun projekti on jotakuinkin rakennettu voi tarkistaa tuotoksiaan laittamalla WampServerin palvelut käyntiin ja tarkastamaan näkymä selaimen kautta. Selaimessa avautui aloitussivuksi määritely dashboard-näkymä. Tässä vaiheessa projektia ei ollut vielä määritelty palvelimen juureksi.



Kuva 14 - Laravel aloitusnäkymä ennen juureksi asettamista

WampServeriin on hyvä määrittää projektin aloitussivu rootiksi eli juureksi. Sen voi tehdä Apachen asetuksista httpd.conf- sekä httpd-vhosts.conf-nimisiin tiedostoihin.

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "${INSTALL_DIR}/www/LaravelProject/public/"
<Directory "${INSTALL_DIR}/www/LaravelProject/public/">
```

Kuva 15 - Apache httpd.conf-tiedosto

```
# Virtual Hosts
#
<VirtualHost *:80>
  ServerName localhost
  ServerAlias localhost
  DocumentRoot "${INSTALL_DIR}/www/LaravelProject/public"
  <Directory "${INSTALL_DIR}/www/LaravelProject/WebApp/public/">
    Options +Indexes +Includes +FollowSymLinks +MultiViews
    AllowOverride All
    Require local
  </Directory>
</VirtualHost>
```

Kuva 16 - Apache httpd-vhost.conf-tiedosto

Laravel-projektin luonnin jälkeen oli vielä tehtävät Git-määritykset, jotta versionhallinta saatiin kuntoon. GitHubiin oli luotu projektia varten oma repositorio, jonka kautta tuotantoonsiirto tapahtui.

5.2 Autentikointi

Laravel 8 ohjelmistokehyksen kanssa voi ottaa käyttöön Laravel Jetstreamin autentikoinnin luomiseksi. Laravel Jetstream on paketti, jonka sisältää palvelinpuolen koodin autentikointia varten ja luo myös tarvittavat käyttöliittymäkomponentit. Laravel Jetstream pitää sisällään Laravel Fortifyn, joka pitää huolen autentikoinnista sekä siihen liittyvistä komponenteista.

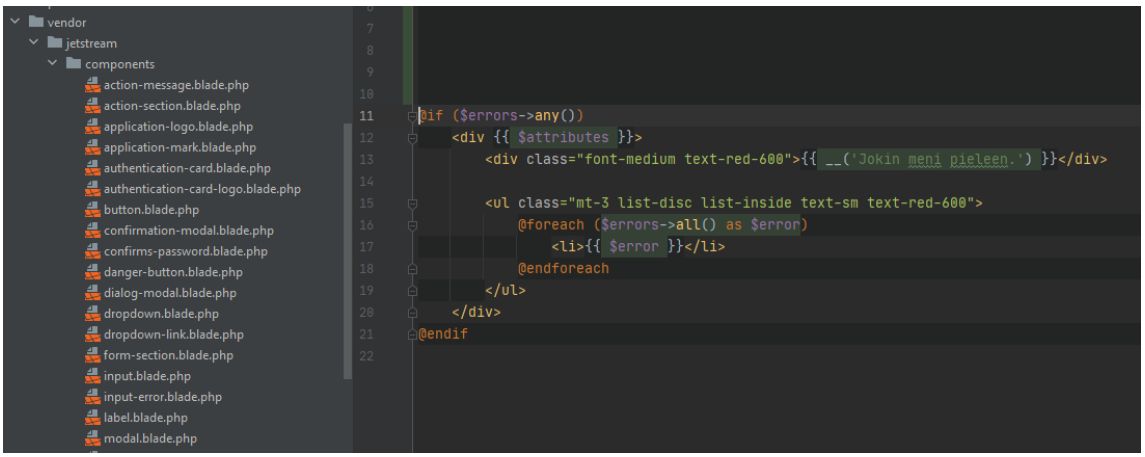
Laravel Jetstream ja Livewire -paketit saa asennettua ajamalla:

```
composer require laravel/Jetstream
php artisan Jetstream:install livewire
```

Pakettien asennuksen jälkeen projektiin on ilmestynyt paljon uusia komponentteja. Esimerkiksi Jetstreamin mukana tulee valtava määrä valmiita näkymäkomponentteja joten näkymiä ei tarvitse luoda itse käsin. Opinnäytetyön aikana huomattiin, että useat komponentit eivät kuitenkaan ole juuri tähän projektiin sopivia. Näkymäkomponentit haetaan projektin vendor-tiedostoista ja niitä ei voida suoraan muokata. Muokkaamista varten kyseiset tiedostot tulee julkaista projektiin. Jetstream-näkymät voidaan julkaista ajamalla komento:

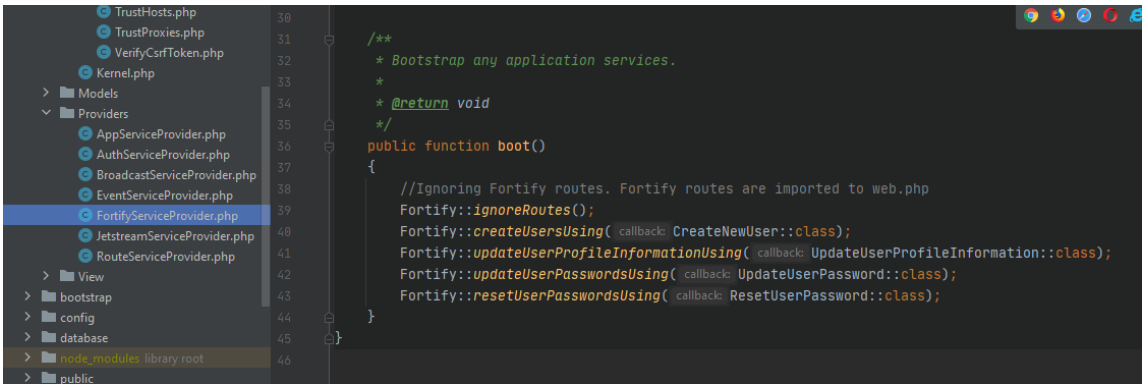
```
php artisan vendor:publish --tag=jetstream-views
```

Komennon ajamisen jälkeen komponentit löytyvät projektin kansiosta View\Vendor\Components ja niitä voidaan muokata vapaasti. Näiden komponenttien tyyleissä käytetään Tailwind.CSS-tyylejä.



Kuva 17 - Jetstream View-tiedosto

Valmiiksi luotuihin autentikointi- ja rekisteröintikomponentteihin jouduttiin tekemään muutoksia myös reitityksen osalta. Haluttu reititys piti muuttaa näkyväksi rekisteröinnin osalta vain kirjautuneelle käyttäjällä, jotta kuka tahansa ei voisi rekisteröityä. Sivuston yksityiset osat suojattiin reitityksessä auth-middlewarella. Rekisteröinnin osalta jouduttiin lisäksi tekemään hieman lisäkonfiguraatioita. Rekisteröinnin taustalla toimii Fortify. Jotta tehty reititys toimisi, täytyy Fortifyn oletusreitit ykset jättää huomioimatta. Se voidaan tehdä FortifyServiceProvider-tiedostossa.



Kuva 18 - FortifyServiceProvider.php tiedosto

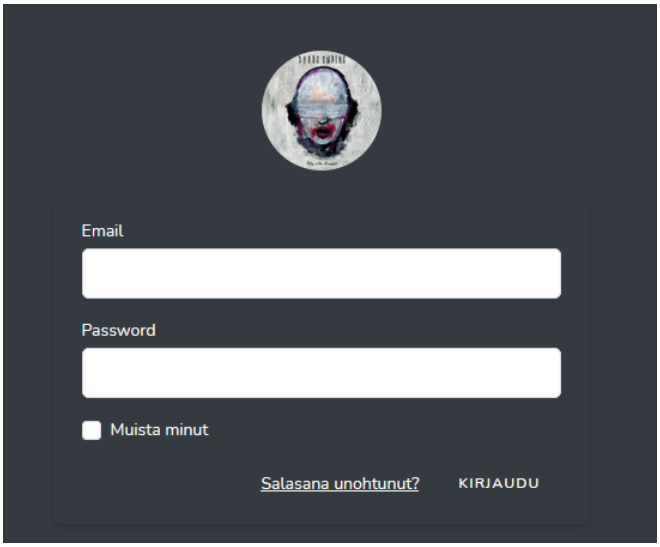
Fortifyn konfiguraatiotiedossa voidaan määritellä mitä ominaisuuksia se käyttää, esimerkiksi tarvittaessa kaksivaiheisen tunnistautumisen. Mikäli haluaa käyttää Laravelin yhteydessä sähköpostivahvistuksia tai sähköpostiviestiä salasanan nollaukseen, täytyy sitä varten luoda sähköpostipalvelin ja lisätä sen tiedot env-tiedostoon.

```
|--------------------------------------------------------------------------
| Features
|--------------------------------------------------------------------------
|
| Some of the Fortify features are optional. You may disable the features
| by removing them from this array. You're free to only remove some of
| these features or you can even remove all of these if you need to.
|
*/

'features' => [
    Features::registration(),
    Features::resetPasswords(),
    Features::emailVerification(),
    Features::updateProfileInformation(),
    Features::updatePasswords(),
    Features::twoFactorAuthentication([
        'confirmPassword' => true,
    ]),
],
```

Kuva 19 - Fortifyn ominaisuuksien määrittely fortify.php-tiedostossa

Fortifyn reititykset ovat ennaltamäärättyjä Fortifyn paikallisissa vendor-tiedostoissa, joita ei tule muuttaa. Tämän sijaan tarvittavat reititykset kopioitiin web.php-tiedostoon, joissa ne muokattiin käyttämään auth-middlewarea.



Kuva 20 – Järjestelmän kirjautumisnäkymä

Pakettien luomisen ja konfiguraatioiden jälkeen käytössä on julkinen sivu kirjautumiselle, ohjelman sisäinen komponentti käyttäjien rekisteröinnille sekä profiilin hallintapaneeli. Profiilin hallintapaneelissa on mahdollista päivittää profiilitietoja, vaihtaa salasana, sulkea avoimia istuntoja muilta laitteilta sekä poistaa tili. Profiilin hallintaan tarkoitettu käyttöliittymä tulee Jestream ja Livewire pakettien mukana. Salasanan validointi toteutetaan Fortifyn PasswordValidationRules-tiedostossa, jonne voidaan määrittää esimerkiksi mitä merkkejä sen tulee sisältää.

5.3 Käyttöliittymä

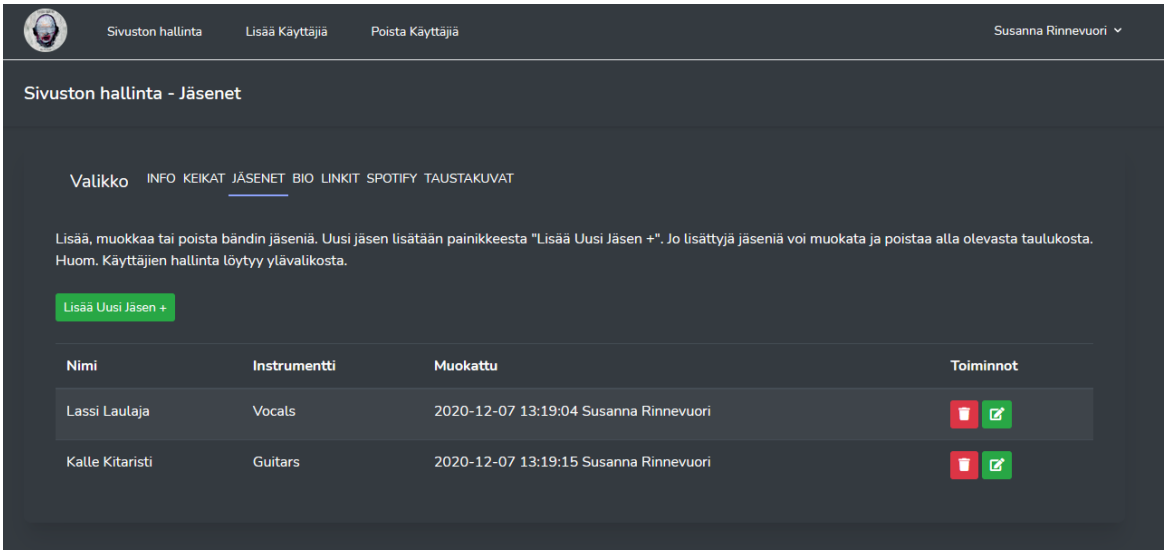
Opinnäytetyön aikana suunniteltiin ja toteutettiin sisällönhallintajärjestelmän osana käyttöliittymä, jonka kautta sivuston omistajat pystyvät hallitsemaan julkisen sivuston osioita. Julkinen sivusto on yhden sivun kokonaisuus eli Single Page Application (SPA) ja käyttöliittymään päätettiin rakentaa niitä vastaavat osiot muiden paitsi taustakuvien osalta. Taustakuvien lisäämiseen ja muokkaamiseen luotiin oma osionsa. Jokaisessa osiossa voidaan toteuttaa tyypilliset CRUD-operaatiot (Create Read Update Delete).

Käyttöliittymätiedostot eli näkymät (View) nimetään Laravelissa blade.php-päätteellä. Blade on tiedosto, joka voi sisältää PHP:tä tai HTML-koodia. Näkymät voidaan palauttaa reitityksen ja ohjaimien (Controller) avulla oheisen esimerkin mukaisesti.

```
//Example route
Route::get( uri: '/users/{user}', [UserController::class, 'show']);
```

Kuva 21 - Reititysesimerkki web.php tiedostossa

Julkinen osa sivustosta tuotiin myös projektiin ja sille tehtiin reititys projektin juureen. Rakenteeseen luotiin omat Blade-tiedostot näkymien osalta kaikille tarvittaville sivuston osioille. Kaikille hallintajärjestelmän sisäisille näkymille asetettiin reitityksessä auth-middleware, jotta näkymä ei olisi julkinen. Lisäksi toteutettiin halutunlainen reititys ja tarvittavat ohjaimet eli controllerit. Sivuston hallinta -valikko on upotettuna muun sivun koodin.

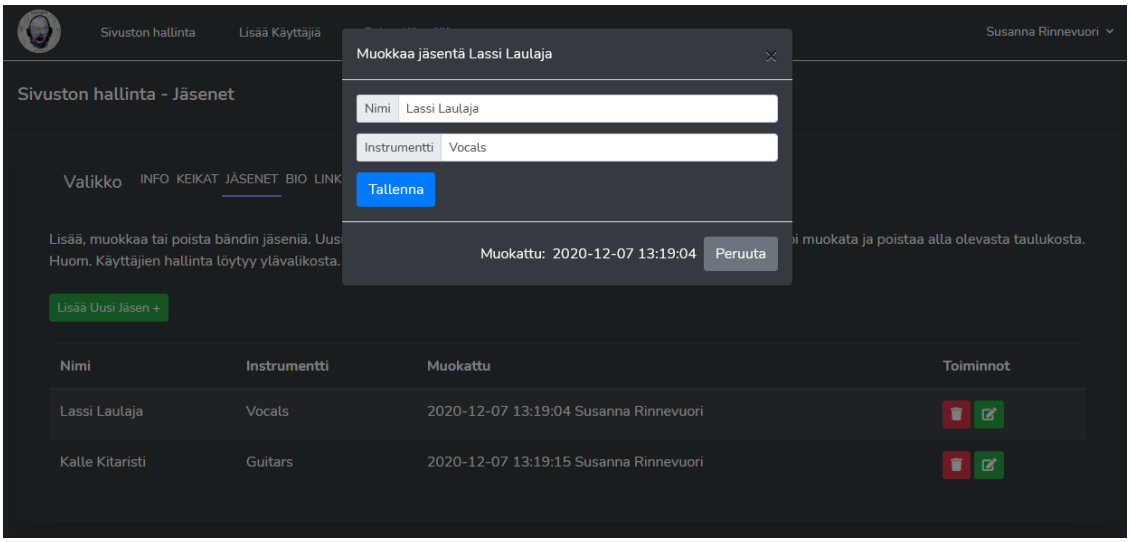


Kuva 22 - Hallintapaneelinäkymä

Käyttöliittymän responsiivisuuden saavuttamiseksi käytettiin Bootstrapia. Bootstrapia voi käyttää joko asentamalla se paikallisesti lataamalla tai pakettimanagerin avulla, mutta se voidaan ottaa myös käyttöön lisäämällä CDN (Content Delivery Network) viittaus koodiin. Jäljempänä olevaa vaihtoehtoa käytettiin tässä opinnäytetyössä.

Bootstrapin lisäksi käytettiin Tailwind.CSS:ää. Tailwind.CSS otettiin käyttöön siksi, että se tuli automaattisesti mukana rakentaessa autentikointia. Koska sivuston niin monet osat käyttivät sitä tyyleissä, jätettiin tyylit paikalleen. Tehtiin vain tarvittavat muutokset värien osalta. Tailwind.CSS:ssä on samalla tavoin luokkia kuin Bootstrapissa.

Uusi sisältö voidaan lisätä sivustolla olevasta painikkeesta ”Uusi”. Jo sivustolle julkaistut sisällöt näkyvät taulukkorakenteissa ja niitä on mahdollista päivittää sekä poistaa. Päivittäessä sisältöä avautuu Bootstrap Modal-komponentti, johon on rakennuttu kentät tietojen päivitystä varten.



Kuva 23 - Bootrap Modalin käyttö

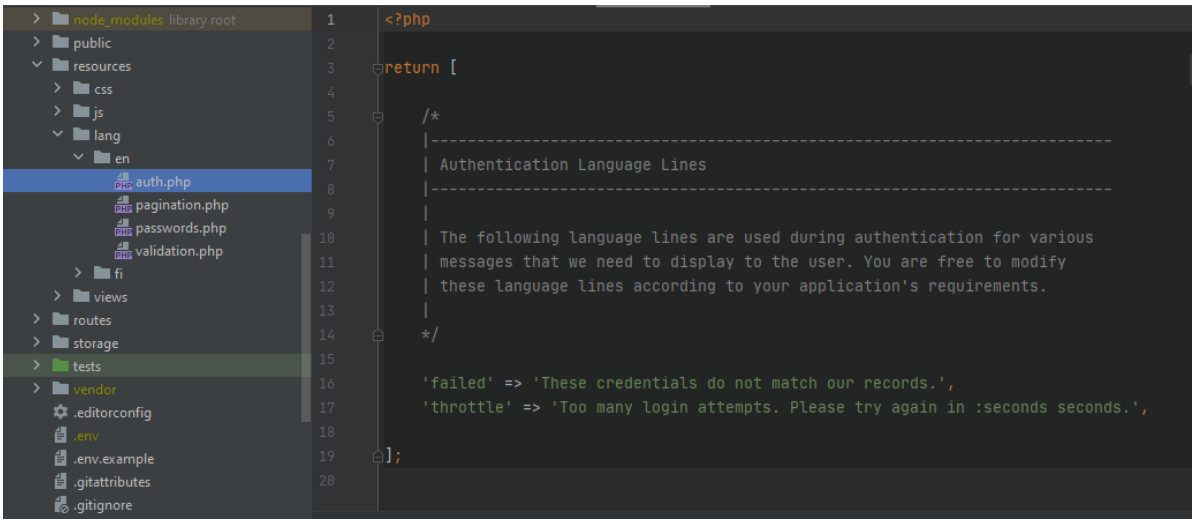
Taustakuville tehtiin hallintapuolelle oma osionsa, josta niitä voidaan hallita. Taustakuvien hallintapaneelissa on hieman erilainen logiikkaa muihin verrattuna.

Hallintakomponentti taustakuvien vaihdolle eroaa opinnäytetyön toteutuksessa hieman muista osista. Palvelimelle tallennetaan vain ne kuvat, jotka ovat käytössä sivustolla eli kaikille sivuston osioille määritetään oma kuva. Uuden kuvan vaihdon yhteydessä vanha kuva poistetaan ja kuvaa lisättäessä kuvan nimeen lisätään aikaleima, jotta samannimistä kuvaa voidaan käyttää halutessa myös muissa kohdissa. Mikäli vanhan kuvan haluaa itselleen talteen, on se mahdollista ladata toisesta painikkeesta. Kuvien lataamiseksi palvelimelle ja palvelimelta ulos käytettiin Laravelin mukana tulevaa Flysystem-pakettia. Jotta kuvien lataaminen olisi mahdollista, tuli kuvien sijainnille antaa internetin kautta pääsy ajamalla artisan-komento:

```
php artisan storage:link
```

Laravelissa on myös mahdollista huomioida lokalisointi ja luoda omia kielipaketteja ohjelman ilmoitusten käyttöön. Autentikoinnin sekä rekisteröinnin komponentit käyttävät lang-kansioon määriteltyjä kieliversioita käyttöliittymän kautta tulevissa ilmoituksissa. Uuden kieliversion luominen tapahtui kopioimalla englannin kielen versiot ja lisäämällä ne uuden kansion alle. Tässä tapauksessa luotiin ’fi’-kansio. Tämän jälkeen kopioituun tiedostoon tehtiin tarvittavat muutokset. Osa käännöksistä ei parametreineen käänny kovin hyvin suomen kielelle ja niiden kanssa jouduttiin joustamaan hieman kieliopillisista asioista. Kielipaketteja käytetään Jetstreamin ja Livewiren yhdistelmän virheilmoituk-

sissa, joten kieliversiot tulevat myös niihin käyttöön. Laravelin avulla valmiiksi luotujen komponenttien staattisia tekstejä tulee muokata niiden näkymätiedostoissa. Ne eivät tule automaattisesti langtiedostoista.



Kuva 24 - Kieliversiot

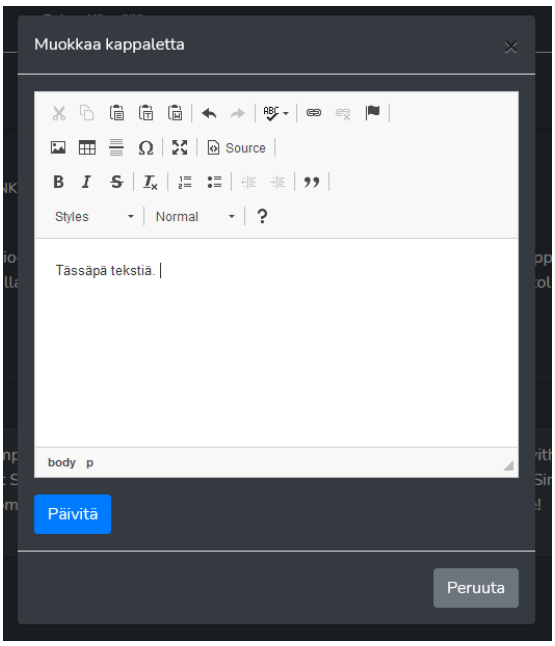
5.3.1 WYSIWYG-editori

Tässä opinnäytetyön kappaleessa käydään läpi WYSIWYG-editoria ja sen osuutta tämän opinnäytetyön aikana. WYSIWYG-editori oli projektin suunnitteluvaiheessa suuremmassa roolissa, mutta se päätettiin kuitenkin jättää pois toteutuksesta.

WYSIWYG on lyhenne englannin kielen sanoista What You See Is What You Get. Tällä tarkoitetaan menetelmää, jolla käyttäjä syöttää jotakin kenttään ja se myös tallennetaan haluttuun kohtaan sellaisenaan. Esimerkiksi voitaisiin syöttää tekstiä, jossa on haluttu muotoilu, kuten erilaisia fonttikokoja eri kohdissa tekstiä sekä vaikkapa jokin kuva. Tallennuksen jälkeen tehty kappale ilmestyisi halutuun kohtaan juuri sellaisena kuin se on hallintamoduulissa tehty.

Opinnäytetyön alkuvaiheessa pohdittiin vaihtoehtoa, jossa WYSIWYG-editorin avulla käyttäjät voisivat tehdä halutut muokkaukset verkkosivustolle nähden samalla lopullisen tuloksen. WYSIWYG-editoreissa hyvinä puolina on se, että siihen voi syöttää muutakin kuin pelkkää tekstiä. Editoriin voi määrittää tekstin olemaan kursiivia, lihavoitua, kappaletekstiä, otsikkotekstiä ja paljon muuta. Tällaisen editorin lisääminen on yleensä helppoa, koska ne ovat Javascript-kirjastoja, eikä näin ollen koodiin tarvitse välttämättä tehdä paljon muutoksia. Riittää, että viittaa kirjastoon ja asettaa komponentin haluttuun kohtaan.

Opinnäytetyön aikana kokeiltiin ottaa CKEditor-niminen WYSIWYG-editori käyttöön. CKEditorin käyttöönottoon on muutamia eri tapoja (CKSource). Tässä käytettiin CDN:ää (Content Delivery Network), eli tarvittavaan pakettiin viitattiin verkon yli. CKEditor upotettiin Bootstrap Modal-komponenttiin. Muokkaustilanteissa Modal-komponenttiin vietiin data sekä CKEditorin tapauksessa, että muissakin Modal-komponenteissa JQueryn avulla. Kantaan viennissä oli huomionarvoista se, että data vietiin HTML-muodossa, mikä ei ole kovin hyvä ratkaisu. Tähän pystyttiin kuitenkin rakentamaan Controllerissa muita tarkistuksia, että koodi on varmasti HTML:ää.



Kuva 25 - CKEditor Bootstrap Modalin sisällä

WYSIWYG-editori saatiin rakennuttua onnistuneesti ja se upotettiin hallintasivuston osioon, jossa syötettiin tekstikappeleita sivustolle. Testauksen aikana havaittiin, ettei siitä ollut hyötyä tällaisessa rakenteessa ja WYSIWYG-editorista päätettiin luopua, vaikka se olikin alusta asti suunnitelmissa toteuttaa. Editorissa oli paljon sellaista, mitä ei sivustolla käytettyihin tekstikappaleisiin tarvittu, kuten kuvien lisääminen sekä otsikkotyylit. Editorin pelkistäminen vain muutamaan toiminnallisuuteen oli lopputuloksen kannalta turhan isotöinen, koska samaan lopputulokseen päästiin myös tavallisella textarea-komponentilla.

```
<div class="form-group">
    <textarea type="text"></textarea>
</div>
```

Kuva 26- Textarea HTML-elementti

Loppuhavaintona editorin osalta todettiin, että käyttöönotto ei ole vaikeaa. Jos haluttaisiin toteuttaa editorin toteutus siten, että se näyttäisi koko sivun, pitäisi tähän nähdä hieman vaivaa. Käytännön ongelmia voi tulla esimerkiksi siinä vaiheessa, kun jo luotua ja sivustolla olevaa tekstiä on paljon. Käyttöliittymän käyttäjäläheisyys ei ehkä tällaisessa tilanteessa ole sitä mitä haluttaisiin. Mikäli sen käyttöön nähdään tulevaisuudessa tarve, toteutus on mahdollista tehdä pienellä vaivalla.

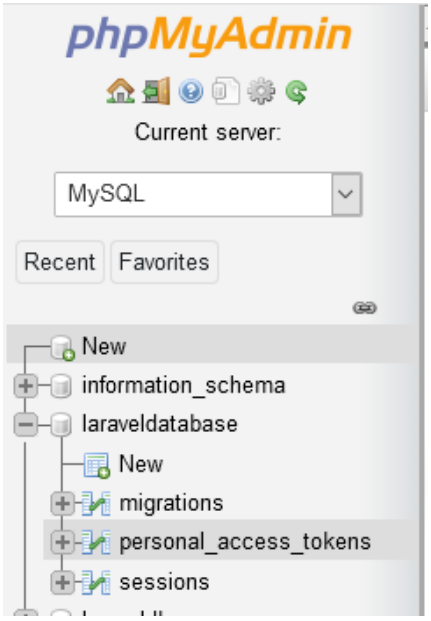
5.4 Tietokanta

Tietokantana käytettiin paikallisessa kehitysympäristössä MySQL-tietokanta. Sen tarkastelu ja hallinta tapahtui WampServerin PhpMyAdminin kautta. PhpMyAdminin kautta luotiin tietokanta ja tietokannan ja Laravel-projektin välistä yhteyttä varten tietokannan tiedot määriteltiin env-tiedostoon.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=databasename
DB_USERNAME=databaseuser
DB_PASSWORD=
DB_CHARSET=utf8
DB_COLLATION=utf8_unicode_ci
```

Kuva 27 - Tietokannan määrittelyesimerkki env-tiedostossa

Tietokannan ja ohjelman välinen keskustelu tapahtuu reitityksen sekä Controller- ja Model-tiedostojen avulla. Laravelin env-tiedoston määritysten kanssa tuli olla tarkkana, että ne vastaavat käytössä olevan tietokannan määrityksiä. Tietokanta luotiin manuaalisesti PhpMyAdmin-käyttöliittymän kautta. Tietokannan tauluilla ei ollut toistensa välisiä yhteyksiä, eli ne toimivat riippumatta toisistaan.



Kuva 28 – Laravel-projektin ensimmäiset migraatiot

Kun tietokanta oli luotu, ei siihen enää tarvinnut koskea. Mikäli tietokantatauluja poistaa käsin, Laravelin tietokantaversionhallinta eli migraatiot menevät sekaisin. Tehdyn migraation voi poistaa, jos migraatioita ei ole vielä ajettu. Jos tietokantataulu on luotu Laravelin migraatioiden kautta, on se esimerkiksi peruttava tai muokattava halutunlaiseksi ajamalla artisan-komento. Opinnäytetyön aikana tietokannan taulut luotiin sekä tarpeen mukaan muokattiin ajamalla tarpeelliset artisan-komennot.

Tietokannan keskustelua varten tuli luoda kantaa vastaavat mallit ja Controllerit. Nämä voi luoda yhtäaikaisesti artisan komennolla

```
php artisan make:model [Name] -controller
```

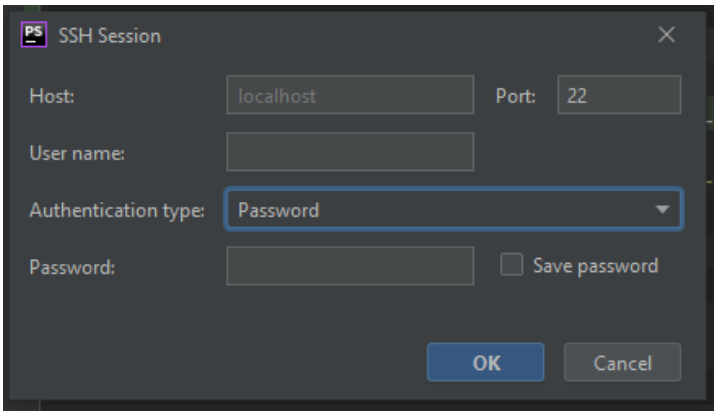
Controller-tiedostoihin luotiin tarpeelliset funktiot tietokannan kanssa keskustelemiseen ja datan palauttamiseksi näkymälle.

```
//Example Controller function
public function getEmail()
{
    $email = DB::table( table: 'users')->where( column: 'name',  operator: 'Matti')->value( column: 'email');
    return $email;
}
```

Kuva 29 - Controller-tiedosto esimerkki

5.5 Tuotantoonsiirto

Opinnäytetyön aikana käytettiin myös paikallisen ympäristön lisäksi myös toista palvelinta testipalvelimena. Tämä palvelin toimi välietappina, jolloin pystyttiin testaamaan, miten sovellus käyttäytyy tuotantoympäristössä. Palvelin oli luotu Linux-ympäristöön ja sinne oli asennettu tyhjä Laravel 8 -projekti. Palvelimeen saatiin SSH-yhteys PhpStormista löytyvän työkalun avulla. Jotta projekti saatiin siirrettyä, täytyi ensin viedä tehdyt muutokset GitHubiin ja palvelimelle kirjautumisen jälkeen hakea tehdyt muutokset palvelimen projektiin.



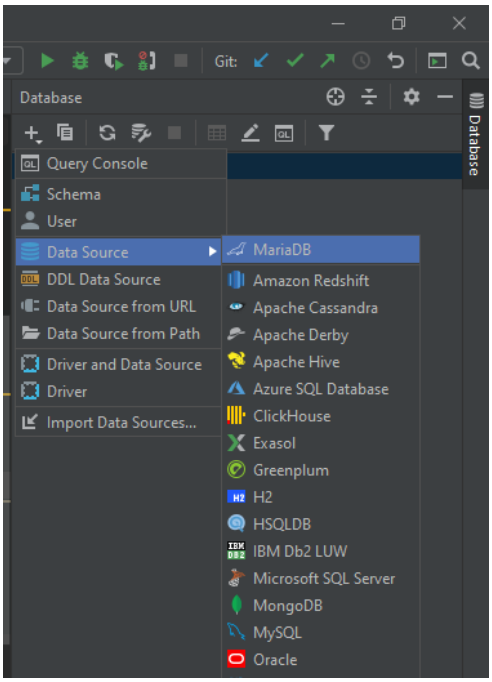
Kuva 30 - SSH-yhteysikkuna PhpStormin kautta

Kun muutokset on viety palvelimelle voi olla tarpeellista ajaa ainakin osa seuraavista komennoista:

php artisan migrate
php artisan config:clear
php artisan cache:clear
php artisan view:clear
php artisan route:clear
composer install
php artisan up

Lisäksi voi olla tarpeen ajaa jotain palvelinkohtaisia komentoja. Yllä mainituista komennoista migraatioiden ajo eli ”php artisan migrate”-komento on tärkeää ajaa tai migraatiot eivät astu voimaan.

PhpStormin kautta sai otettua myös yhteyden erilliseen tietokantaan ”Database”-valikosta valitsemalla halutun tietokannan ja syöttämällä tarvittavat tiedot yhteyttä varten.



Kuva 31 – Tietokantayhteyden ottaminen PhpStormin kautta

6 POHDINTA

PHP sanotaan olevan ohjelmistokehittäjien parissa kuoleva kieli, mutta tällaisilla ohjelmistokehyksillä voidaan pelastaa paljon. PHP:n ongelma saattaa olla enemmänkin siinä, että se voi olla ensimmäinen kosketus koodaamisen maailmaan Wordpressin kautta. PHP ei välttämättä ole ihanteellinen kieli aloittaa ohjelmointia, mikä ei todennäköisesti ole selvää monellekaan Wordpressin kautta ohjelmointiin tutustuvalle. Opinnäytetyön aihe oli mielenkiintoinen ja se opetti paljon uutta websovelluskehityksestä, etenkin backendistä sekä Laravel-ohjelmistokehyksestä. Laraveliin tutustamalla tuli tunne, ettei PHP välttämättä ole vielä kuolemassa. Toivottavasti esimerkiksi Wordpress ja muut suuret PHP:tä käyttävät yritykset pystyisivät ottamaan jonkinlaisia integraatioita nykyaikaisemmista tekniikoista käyttäjiensä saataville helppokäyttöisyys edellä.

Opinnäytetyön tuotoksena syntyi toimiva kokonaisuus sisällönhallintaan Single Page Application -sivustolle, joka rakennettiin käyttäen Laravelin versiota 8. Muita tekniikoita oli HTML5, CSS, Javascript sekä JQuery.

Valmiissa järjestelmässä kunkin osion sisältöjä voidaan luoda, tarkastella, päivittää sekä poistaa eli niille voi tehdä CRUD-operaatiot. Autentikointi ja rekisteröinti luotiin Laravelin valmiilla komponenteilla ja valmiissa komponenteissa käytettiin tyyleissä apuna Tailwind.CSS:ää. Laravelin MVC-mallia (Model View Controller) on helppo ymmärtää ja erilaiset komponentit tekevät sivuston rakentamisesta helppoa. Tämä kuitenkin vaatii dokumentaatioon perehtymistä, joka on onneksi erittäin hyvä. Virheitä kuitenkin saattaa tulla melko helposti, joten dokumentaatioon on syytä lukea tarkasti. Opinnäytetyön aikana eteen tulikin muutamia ongelmia, jotka johtuivat suurimmaksi osaksi huonosta dokumentaation lukemisesta.

Projektin edetessä vastaan tuli Tailwind.CSS, joka tuli Laravelin mukana automaattisesti. Tämän lisäksi tarvittaessa käytettiin Bootstrapia, koska sen käytöstä on kokemusta. Olisi kuitenkin saattanut olla mielenkiintoista rakentaa sovelluksen tyylit käyttäen ainoastaan Tailwind.CSS:ää.

Opinnäytetyön aikana nousi esiin joitakin asioita, joita mielestäni voisi vielä jalostaa ja jatkokehittää. Sivuston rakenne on edelleen staattinen, siellä on tietty määrä osioita, eikä niitä voida tällä hetkellä sisällönhallintajärjestelmän kautta lisätä. Tämän lisäksi sivun osion rakenne on ennalta määrätty. Tähän voisi rakentaa jotain dynaamisempaa ja on varmasti toteutettavissa riittävän ajan puitteissa. Käyttöliittymän valikoissa voisi myös olla korjattavaa rakenteen osalta, mutta muutoin lopputulos oli melko onnistunut.

LÄHTEET

Automattic. Wordpress.com. [Online] [Viitattu: 10. 12 2020.] https://wordpress.com/create-website/.

Bourdon, Romain. WampServer. [Online] [Viitattu: 10. 12 2020.] https://www.wampserver.com/en/.

CKSource. CKEditor Ecosystem Documentation. [Online] [Viitattu: 01. 12 2020.] https://ckeditor.com/docs/ckeditor4/latest/.

Code Boxx. Code Boxx. [Online] [Viitattu: 10. 12 2020.] https://code-boxx.com/difference-wamp-lamp-mamp-xampp/.

Composer. Composer. [Online] [Viitattu: 08. 12 2020.] https://getcomposer.org/doc/00-intro.md.

GitHub. GitHub. [Online] GitHub Inc. [Viitattu: 02. 10 2020.] https://github.com/.

GitHub, Inc. 2020. GitHub Guides. [Online] 24. 07 2020. [Viitattu: 05. 12 2020.] https://guides.github.com/activities/hello-world/.

Ionos. Ionos. [Online] [Viitattu: 10. 12 2020.] https://www.ionos.com/digitalguide/hosting/cms/cms-comparison-a-review-of-the-best-platforms/.

Ite Wiki. Ite Wiki. [Online] [Viitattu: 06. 12 2020.] https://www.itewiki.fi/opas/content-management-system/.

Jetbrains. PhpStorm. [Online] [Viitattu: 09. 12 2020.] https://www.jetbrains.com/phpstorm/.

Laravell LLC. Laravel. [Online] Laravel LCC. [Viitattu: 02. 10 2020.] https://laravel.com/docs/4.2/introduction.

Light IT. Ligth IT. [Online] [Viitattu: 20. 10 2020.] https://light-it.net/blog/why-use-php-main-advantages-and-disadvantages/.

Medium. Medium. [Online] [Viitattu: 06. 12 2020.] https://medium.com/@KNOWARTH/how-to-choose-a-content-management-system-cms-for-your-site-9bfc483a5fc0.

Statista Inc. Statista. [Online] [Viitattu: 30. 11 2020.] https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/.

Taylor Otwell. Laravel. [Online] [Viitattu: 15. 12 2020.] https://laravel.com/docs/8.x/releases.

—. Laravel. [Online] [Viitattu: 1. 12 2020.] https://laravel.com/docs/8.x/blade.

—. Laravel. [Online] [Viitattu: 10. 12 2020.] https://laravel.com/docs/8.x/csrf#csrf-introduction.

The PHP Group. php.net. [Online] [Viitattu: 16. 12 2020.] https://www.php.net/supported-versions.php.