LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Sisällönhallintajärjestelmän suunnittelu ja toteutus Laravelilla

- Case Nuuksionmaja

Mäkinen, Lauri

2016 Laurea

Laurea-ammattikorkeakoulu

Sisällönhallintajärjestelmän suunnittelu ja toteutus Laravelilla - Case Nuuksionmaja

Mäkinen, Lauri
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Toukokuu, 2016

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma

Mäkinen, Lauri

Sisällönhallintajärjestelmän suunnittelu ja toteutus Laravelilla - Case Nuuksionmaja

Vuosi 2016 Sivumäärä 33

Tämän opinnäytetyön tarkoituksena oli uudistaa toimeksiantajana olleen Nuuksionmajan juhlatilan verkkosivusto käyttäen nykyaikaisia web-kehityksen työkaluja. Tavoitteena oli kehittää asiakkaan toivomusten mukainen sisällönhallintajärjestelmä, jonka avulla asiakas pystyisi helposti itse ylläpitää sivuston sisältöä.

Valmiita sisällönhallintajärjestelmiä on verkossa saatavissa useita eri vaihtoehtoja. Tässä opinnäytetyössä sisällönhallintajärjestelmä haluttiin kuitenkin toteuttaa alusta alkaen itse, jotta lopputulos olisi mahdollisimman lähellä toimeksiantajan toiveita sivustoon liittyen. Tästä syystä verkkosivusto päätettiin kehittää käyttäen avoimen lähdekoodin Laravel ohjelmistokehystä. Laravel tarjosi kehittäjälle työkalut luoda moderni, tietoturvallinen ja käyttäjäystävällinen verkkosivusto.

Toiminnallisen opinnäytetyön tuloksena Nuuksionmajalle luotiin helppokäyttöinen ja asiakkaan toiveita vastaava verkkosivusto, jossa asiakas pystyy itse ylläpitämään sivuston sisältöä.

Asiasanat: Laravel, sisällönhallintajärjestelmä, PHP, web-kehitys

Laurea University of Applied Sciences **Abstract**
Bachelor's Degree Programme in Business Information Technology

Mäkinen, Lauri

Planning and Implementation of a Laravel Based Content Management System – A Case study of Nuuksionmaja

Year 2016 Pages 33

The purpose of this thesis was to recreate Nuuksionmaja's old website using modern web development tools. The goal was to create a user-friendly content management system based on the customer's needs, following the old outlook of Nuuksionmaja's previous website. Based on the customer's needs, one of the most important parts of this process was to make it so that the customer could update the content of the website on his/her own and thus save on the expenses related to the website.

The internet is full of ready-made, popular and free content management systems. However, for the purposes of this thesis, a raw PHP framework, Laravel, was chosen as the platform to develop the website on. Using a raw framework forces the developer to learn more about the development process itself and create a unique experience for the customer. Laravel provides all the necessary building blocks needed to create a modern, secure and user-friendly website.

As the product of this thesis, Nuuksionmaja received a fully working, production ready website, in which the customer could update the content of the website on his/her own.

Keywords: Laravel, Content Management System, PHP, Web-development

Sisällys

1 Johdanto

Internet näkyvyys on yrityksille nykyään hyvin tärkeässä roolissa. Riippuen tietysti yrityksen toimialasta, voi internet-sivusto olla jopa sen ainoa tapa luoda yritykselle tulosta. Jotta yritys pystyisi kilpailemaan tasavertaisesti muiden alalla toimivien yritysten kanssa, on siis oltava ajan hermolla ja panostettava helppokäyttöiseen, informatiiviseen ja toimivaan internet-sivustoon.

1.1 Lähtökohdat ja tavoitteet

Tämän opinnäytetyön tavoitteena on luoda Nuuksionmajalle helppokäyttöinen, informatiivinen ja toimiva verkkosivusto. Käytön helppous ei saa rajoittua ainoastaan asiakkaisiin, vaan myös sivuston ylläpitäjään, eli itse yrittäjään. Tästä syystä sivustolle suunnitellaan ylläpito-osio, jonka avulla asiakas pystyy itse helposti ylläpitämään sivuston sisältöä miellyttävän käyttöliittymän välityksellä. Ylläpito-osio vaatii kirjautumisen järjestelmään ja on näin ollen nähtävissä ainoastaan kirjautuneille käyttäjille.

Nuuksionmajalla oli ennestään verkkosivusto, jota asiakas itse ei pystynyt päivittämään. Vanha verkkosivusto oli luotu staattiseksi, joka tässä tapauksessa pakotti asiakkaan aina päivityksen yhteydessä olemaan yhteydessä sivuston kehittäjään. Tästä aiheutuneet kulut olivat erittäin suuret, joka lopulta johti sivuston tiedon vanhenemiseen ja sivuston taka-alalle jäämiseen.

1.2 Aiheen rajaus

Jotta opinnäytetyön laajuus ei leviäisi liian suureksi, on sen aihetta rajattava. Tässä opinnäytetyössä käsiteltävät aiheet keskittyvät lähinnä sivuston tekniseen toteutukseen ja siitä nouseviin oivalluksiin. Tämän opinnäytetyön tarkoituksena ei myöskään ole opettaa PHP-ohjelmointia, joten lukijan oletetaan ymmärtävän vähintään PHP:n olio-ohjelmoinnin perusteet.

Opinnäytetyössä luotu sivusto toteutettiin avoimen lähdekoodin Laravel PHP-ohjelmistokehyksellä (framework) ja MySQL-arkkitehtuuriin perustuvalla MariaDB-tietokannalla. Työssä tarkastellaan sitä, miten Laravelilla voidaan luoda helposti päivitettävä asiakkaan toiveita ja tarpeita vastaava verkkosivusto.

1.3 Toimeksiantaja

Opinnäytetyössä luotava sivusto kehitettiin Espoon Nuuksiossa sijaitsevalle Nuuksionmajan juhlatilalle. Nuuksionmaja on Espoon Nuuksiossa sijaitseva 1800-luvun lopulla rakennettu perinteikäs kartanomainen rakennus, jonka tiloissa asiakkaat voivat järjestää

juhlatilaisuuksia. Pääasiassa majalla järjestetään häätilaisuuksia, syntymäpäiväjuhlia ja koulutustapahtumia. Nuuksionmajalle verkkosivusto on erittäin tärkeä osa sen toimintaa, koska sivustolla majan isäntä voi esitellä mm. tarjolla olevia menuehdotelmia, antaa lisätietoa majasta ja tarjota mahdollisuuden yhteydenottoon.

2 Menetelmät ja työkalut

Tätä opinnäytetyötä tehdessä olivat käytetyt menetelmät ja työkalut erittäin suuressa roolissa. Sivustojen suunnittelu ja kehittäminen ovat molemmat erittäin työkalusidonnaisia toimia, joissa työn lopputulos on suuresti riippuvainen käytetyistä työkaluista ja menetelmistä. Tulevassa luvussa käsitellään opinnäytetyössä käytettyjä tutkimusmenetelmiä ja työkaluja.

2.1 Toiminnallinen opinnäytetyö

Opinnäytetyö toteutettiin toiminnallisena opinnäytetyönä, jossa pyritään esittämään käytännönläheisesti sivuston kehityksen vaiheet. Airaksisen (2009) mukaan toiminnallinen opinnäytetyö on yksi ammattikorkeakoulun opinnäytetyön muodoista, jossa tavoitellaan käytännön toiminnan ohjeistamista, opastamista, järjestämistä ja järkeistämistä.

Tätä opinnäytetyötä varten toiminnallinen opinnäytetyö oli selvä valinta, koska toimeksiantaja tarvitsi uuden verkkosivuston. Opinnäytetyö toimii jatkossa opasteena samanlaisia web-sovellusta kehittäville kehittäjille ja todisteena työn tuloksista toimeksiantajalle. Koska opinnäytetyön halutaan toimivan jatkossa myös opasteena muille kehittäjille, on tässä opinnäytetyössä nähtävissä hyvin paljon koodi esimerkkejä.

2.2 Responsiivinen web-suunnittelu

Nykyään verkkosivustojen käyttäjät käyttävät yhä enemmän mobiililaitteita sivustojen tarkasteluun, joka pakottaa kehittäjät vastaamaan käyttäjien tarpeisiin. Responsiivisen web-suunnittelun (mobiililaitteita varten suunniteltu) tarkoituksena on vastata käyttäjän tarpeita kaikissa tilanteissa. Verkkosivuston tulisi siis olla helppokäyttöinen ja informatiivinen näytön koosta riippumatta.

Responsiivinen web-suunnittelu vastaa mobiililaitteiden web-sovelluksille asettamiin vaatimuksiin mukauttamalla sivuston sisällön näytön kokoon sopivaksi. Yleensä responsiivisessa web-suunnittelussa sivuston ulkomuodolle asetetaan tiettyjä pikselimääräisiä pisteitä (break point), joissa sivuston ulkomuoto muuttuu esim. päätteen leveydelle sopivaan muotoon. Yleisin tapa tehdä tämä on se, että tietokoneen ruudulla sivusto on leveämpi, kuin esim. tabletilla tai puhelimella. Sivuston sisältö siis mukautuu käyttäjän tarpeisiin

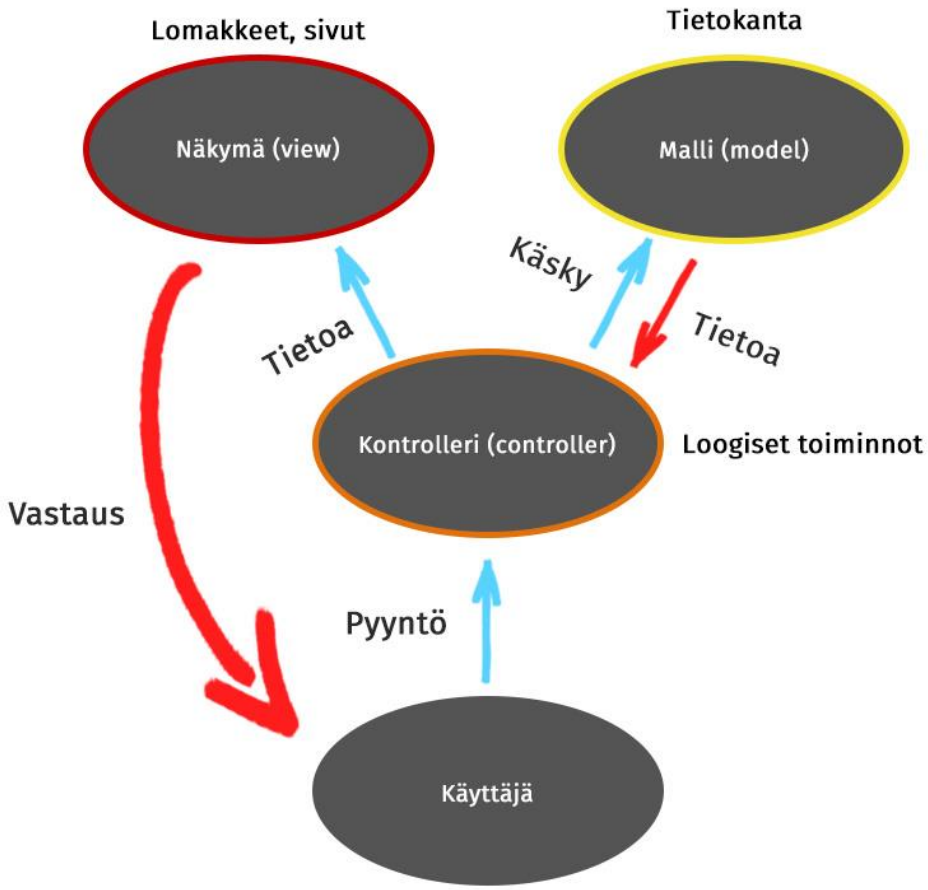
muuttamalla ulkomuotoaan ja mahdollisesti myös esitystapaansa, jotta tarpeellinen tieto saadaan käyttäjälle mieleiseen muotoon.

2.3 Avoimen lähdekoodin ohjelmistot

Avoimeen lähdekoodiin (engl. open source) perustuvien ohjelmistojen lähdekoodi on nimensä mukaisesti kaikille avointa, vapaasti muokattavaa ja käytettävää. Perinteisesti ohjelmistoja on kehitetty niin, että ohjelmiston lähdekoodi ei ole muiden kuin kehittäjien nähtävissä, mutta viime vuosikymmeninä suosituksi noussut avoimen lähdekoodin lähestymistapa on kiihdyttänyt ohjelmistojen koodin laatua ja tehokuutta.

2.4 MVC-ohjelmistoarkkitehtuuri

MVC (Model, View, Controller) on ohjelmistoarkkitehtuuri, joka pyrkii selkeästi erottamaan sovelluksen toiminnollisuuden sen käyttöliittymästä. Dangarin (2013, 8) mukaan MVC-arkkitehtuurin oleellisin erottava tekijä on se, että sovelluksen logiikka erotetaan selkeästi sen esityskerroksesta. Opinnäytetyössä käytettävä Laravel-ohjelmistokehys perustuu MVC-arkkitehtuuriin, ja näin ollen olettaa myös kehittäjän seuraavan sen mallia. MVC-arkkitehtuurissa model (malli), view (näkymä) ja kontrolleri (controller) jaetaan erillisiin tiedostoihin, jolloin ohjelmiston rakenteesta saadaan hyvin selkeä kokonaisuus.

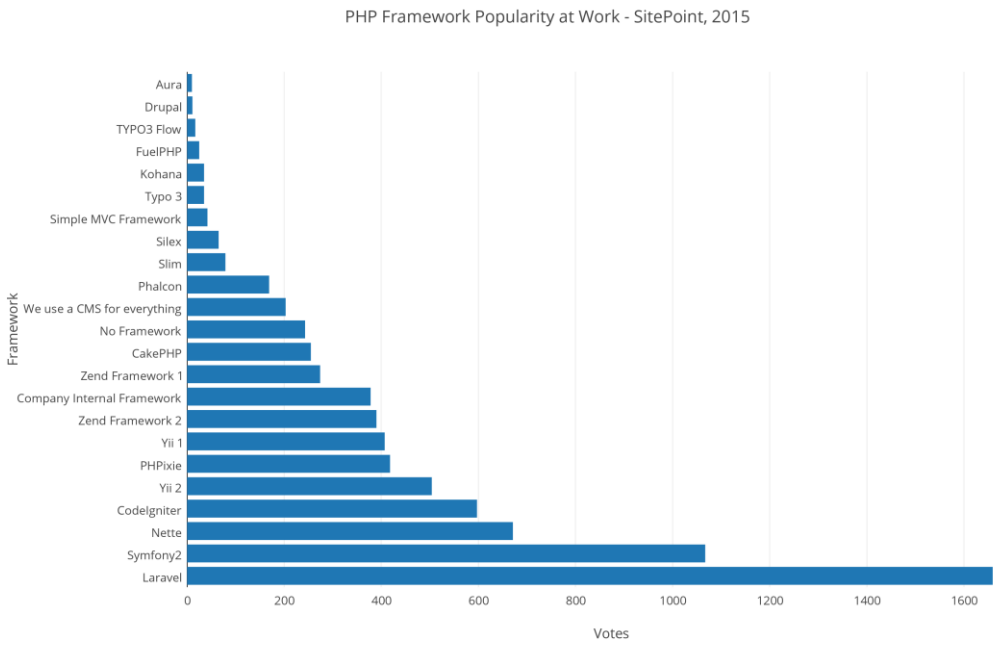


Kuvio 1: MVC-arkkitehtuurin toiminta

MVC on ollut laajalti käytössä jo jonkin aikaa niin ohjelmisto- kuin web-kehityksessäkin. Dangari (2013, 8) mainitsee luvussa Landing yourself into the MVC world, että MVC on löydettävissä lähes kaikkialta. Ohjelmistokehityksessä MVC-arkkitehtuuri koetaan yleisesti hyvin selkeänä ja helposti testattavana ratkaisuna. Huonoja puolia arkkitehtuurissa ei yleisesti koeta olevan kovin monia. Oikeastaan ainoana huonona MVC:n puolena voidaan pitää sitä, että sovellukset ovat hieman työläämpiä kehittää, kuin perinteisellä yhden tiedoston tyylillä luotavat sovellukset.

2.5 Laravel-ohjelmistokehys

Laravel on Taylor Otwellin vuonna 2011 kehittämä avoimen lähdekoodin MVC-pohjainen PHP-ohjelmistokehys. Laravel on julkaisunsa jälkeen saanut useita päivityksiä ja opinnäytetyön kirjoittamisen aikaan kehyksen viimeisin versio on 5.2.15, joka julkaistiin 12.2.2016. Laravel on yksi maailman suosituimmista PHP-ohjelmistokehyksistä. Tämän suosion tuloksena avoimeen lähdekoodiin perustuvaa Laravelia on ollut mukana kehittämässä useita satoja PHP-ohjelmoijia. Saunierin (2014, 12) mukaan Laravel on pohjimmiltaan yksinkertainen ja hienosti itseään ilmaiseva PHP ohjelmistokehys, jonka voi huomata mm. yksityiskohtien kautta.



Kuvio 2: SitePointin PHP-framework-kyselyn tulokset

Laravel antaa kehittäjille tehokkaita työkaluja esim. tietokantayhteyksien luomista, sovelluksen turvallisuutta, ulkoista esitystapaa ja sovelluksen päivittämistä varten.

2.6 Bootstrap-ohjelmistokehys

Bootstrap on CSS- ja Javascript-kehys, joka antaa kehittäjille tehokkaan tavan luoda myös mobiililaitteille sopivia sovelluksia. Bootstrapin avulla sovellusten ulkoasu voidaan tarpeen mukaan jakaa riveihin ja sarakkeisiin, jolloin ulkoasu pysyy selkeänä ja yhtenäisenä. Samalla ulkoasu voidaan jäsentää eri tavalla päätteen koosta riippuen.

Tässä projektissa käytettiin Bootstrapin versiota 3.3.6. Bootstrapilla mahdollistettiin nopea sivuston ulkoasun suunnittelu ja kehitys. Samalla kehys varmistaa sen, että sivuston kaikki osa-alueet ovat laadullisesti tasaisia.

2.7 Git versionhallinta

Git on hajautettu versionhallintajärjestelmä, jossa jokaisella kehittäjällä on oma versio sovelluksen koodipohjasta. Somasundaram:n (2013, 8) määrittelee versionhallintajärjestelmän järjestelmäksi, joka pystyy tallentamaan muutoksia tiedostoihin niin lyhyen, kuin pitkänkin aikavälin aikana ja aina olemaan valmis palauttamaan vanhemman version tiedostosta, jos tarve niin vaatii.

Git on itsenäinen versionhallintajärjestelmä, joten se ei tarvitse ulkopuolista palvelua toimiakseen. Jos käyttäjä siis haluaa, voi hän itse tallentaa versioimansa tiedostot mihin tahansa. Tiedostojen jakamista varten on kuitenkin edullista käyttää esim. GitHub, GitLab tai Bitbucket Git säilytyspaikka palveluita. Edellä mainitut palvelut tarjoavat palvelimiltaan tilaa säilyttää Git-versionhallinan alaiset tiedostot. Näin esimerkiksi sovelluksen siirtäminen kehitysympäristöstä tuotantoon, tai tiedostojen jakaminen tiimin kesken on huomattavasti helpompaa.

Tässä opinnäytetyössä Git-versionhallintajärjestelmää käytettiin hallitsemaan sovelluskehityksen eri vaiheita ja esittelemään uusia ominaisuuksia asiakkaalle. Git:n avulla tiedostot myös siirrettiin kehitysympäristöstä tuotantopalvelimelle. Projektin tiedostot tallennettiin Bitbucket-palveluun.

2.8 Laravel Homestead kehitysympäristö

Koska sivusto kehitetään PHP:lla, tarvitaan kehitysympäristöön myös PHP:n esikääntäjä. Laravelin kehitystiimi on luonut Homestead-nimisen virtuaaliympäristön, jossa mm. Laravel-pohjaisia web-sovelluksia on erittäin nopea ja helppo kehittää. Laravel Homestead perustuu Vagrant-nimiseen virtuaali kehitysympäristöjä luovaan ohjelmistoon. Vagrant toimii mm. seuraavien virtualisointiohjelmistojen kanssa: VirtualBox, VMWare, KVM ja LXC.

Virtualisoimalla kehitysympäristö, voidaan varmistua siitä, että tulevalla tuotantopalvelimella käytettävät ohjelmistoversiot vastaavat kehityksessä käytettäviä ohjelmisto versioita. Näin vältytään tuotantoon siirryttäessä ongelmalta, jossa jokin ohjelmiston osa-alue ei toimi tuotannossa odotetulla tavalla, koska esimerkiksi kehitysympäristön ja tuotannon ohjelmistot eivät vastaa toisiaan.

3 Sivuston suunnittelu ja mallintaminen

Sivuston suunnittelu pohjautui toimeksiantajan toiveesta edellisen sivuston ulkoasuun. Edellisen sivuston rakenne ei kuitenkaan ollut pohjimmiltaan responsiivinen, joten suunnitteluvaiheessa jouduttiin tekemään melko suuria muutoksia. Tietokantaa mallinnettaessa käytettiin hyväksi Laravel Schema Designer-työkalua, jonka avulla pystyttiin ennen toteutusta suunnittelemaan tietokannan rakenne.

3.1 Haastattelu ja vaatimusmäärittely

Opinnäytetyön projektia varten toimeksiantajan kanssa käytiin haastattelun muodossa läpi ne asiat, joita verkkosivustolla toivottiin olevan ja miltä sen toivottiin näyttävän. Haastattelun aikana kävi ilmi, että vanhan verkkosivuston ulkoasuun oltiin jo ennestään tyytyväisiä.

Tulevan sivuston kuitenkin toivottiin olevan käytettävissä myös mobiililaitteilla. Vanha verkkosivusto ei valitettavasti ollut responsiivinen, joten sivuston ulkomuotoa jouduttaisiin jonkin verran muokkaamaan. Suunnittelussa ja toteutuksessa kuitenkin pyrittäisiin pitämään vanhan sivuston lämmin ja maanläheinen ulkoasu.

Kuvio 3: Nuuksionmajan vanha sivusto

Toiminnollisuudeltaan uudelle verkkosivustolle haluttiin osio viimeisimmille uutisille, kuvagalleria, yhteydenottolomake ja tietoa Nuuksionmajasta. Mahdollisuuksien mukaan kaiken sisällön toivottiin olevan muokattavissa asiakkaan toimesta, mutta erityisesti kiinnitettiin huomiota siihen, että Majalla tarjottavat menut, hinnasto ja kuvagallerian kuvat olisivat asiakkaan muokattavissa.

3.2 Toiminnollisuuden ja ulkoasun suunnittelu

Haastattelun ja vanhan sivuston tiedon perusteella verkkosivun ulkoasusta luotiin alustava suunnitelma. Suunnitelma luotiin ensin kuvankäsittelyohjelmalla, joka lähetettiin asiakkaalle hyväksyttäväksi. Kun asiakas oli tyytyväinen suunnitelman ulkomuotoon, luotiin kuvan perusteella HTML, CSS ja Javascript esimerkki sivustosta. Esimerkki lähetettiin palvelimelle asiakkaan tarkasteltavaksi. Toimeksiantaja oli erittäin tyytyväinen näkemäänsä, joten tämän jälkeen päästiin suunnittelemaan sovelluksen toiminnallisuutta.

Kuvio 4: Uuden sivuston alustava suunnitelma

Toiminnollisuuden puolesta ohjelmistokehityksessä sovellus- ja tietokantarakenteen suunnittelu ovat erittäin tärkeitä osia sovelluskehitystä. Koska asiakas toivoi, että suuri osa sivuston sisällöstä tulisi olla heidän muokattavissaan, pyrittiin lähes kaikki sivustolla näkyvä tieto tallentamaan tietokantaan.

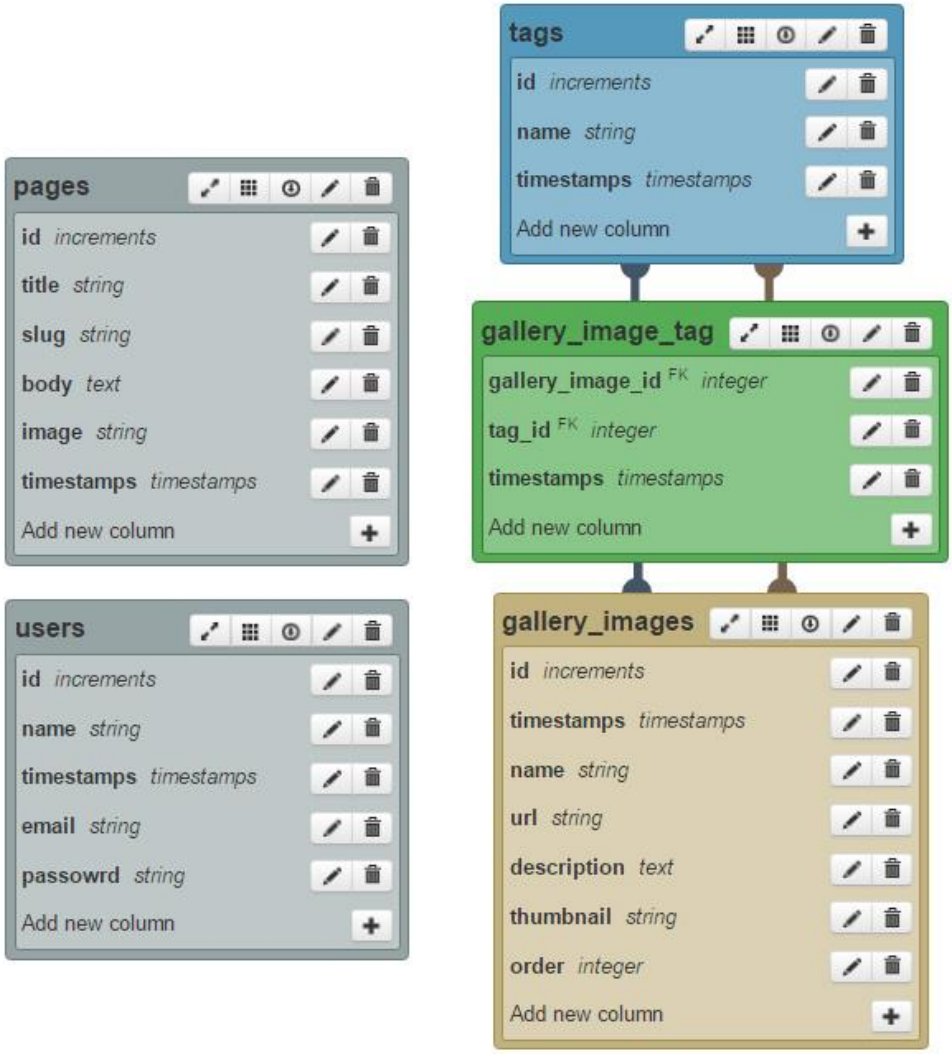
Tietokannan suunnitellessa käytettiin Laravel Schema Designeria, jonka on tarkoitus nopeuttaa ja helpottaa tietokantojen suunnittelua erityisesti Laravelia käytettäessä. Tietokantaa suunniteltaessa Laravel Schema Designer luo tietokannan mallin lisäksi valmiit MVC-malliin perustuvat modelit ja modelien yhteydet. Suunnitelma tietokannan rakenteesta-kuviossa voidaan nähdä, miten tags ja gallery_images taulut on yhdistetty toisiinsa ja näiden taulujen väliin on luotu yhdistävä pivot-taulu, jossa viitataan molempien taulujen ID-kenttiin. Tämä mahdollistaa modelien toisiinsa yhdistämisen ja niiden helpon kutsumisen koodissa.

Modelit	**Viewit**	**Controllerit**

Page, GalleryImage, Tag	Käsitellään tarkemmin tekstissä.	PagesController, GalleryController, UploadController

Taulukko 1: Toteutusvaiheessa luotavat MVC-mallin mukaiset elementit

Laravelin tietokantaa suunnitellessa tietokantaan ei tarvitse erikseen luoda käyttäjille omaa tauluaan, koska ohjelmistokehyksessä tulee oletuksena mukana taulu käyttäjiä varten. Suunnitelmaan tämä taulu on kuitenkin sijoitettu, jotta sovelluksen tietokannasta saadaan mahdollisimman hyvä kuva.



Kuvio 5: Suunnitelma tietokannan rakenteesta

4 Sivuston toteutus

Tämän sivuston kehittämiseen käytettiin Windows 10 käyttöjärjestelmää, eikä tässä luvussa tulla käsittelemään muiden käyttöjärjestelmien toimintatapoja.

Vaikkakin suunnittelun jälkeen tämän sovelluksen kehittäminen aloitettiin Laravel Homestead-virtuaalikoneen asentamisella ja sen asetusten määrittelemisellä, tämän opinnäytetyön rajauksen vuoksi, en tässä osiossa keskity kuvaamaan Laravel Homesteadin asentamista. Laravel Homestead ei myöskään ole välttämätön Laravel sovelluksia kehitettäessä, vaan mikä vain PHP-tulkilla (ja Laravelin kannalta tarpeellisilla PHP-lisäosilla) varustettu paikallinen palvelin kelpaa. Ennen virtualisoinnin yleistymistä, käyttivät PHP-kehittäjät usein esim. Apache Friends:n XAMPP-ohjelmistoa X (cross-platform), Apache, MySQL, PHP, Perl simuloidakseen tulevaa palvelinympäristöä.

4.1 Alustava asennus - Composer

Virtuaalikoneen asentamisen jälkeen täytyy isäntäkoneelle asentaa Composer PHP-pakettien hallintaohjelma. Composerin asentaminen onnistuu seuraamalla Composerin asennusohjeita osoitteessa: https://getcomposer.org/download/ (Composer.org 2016).

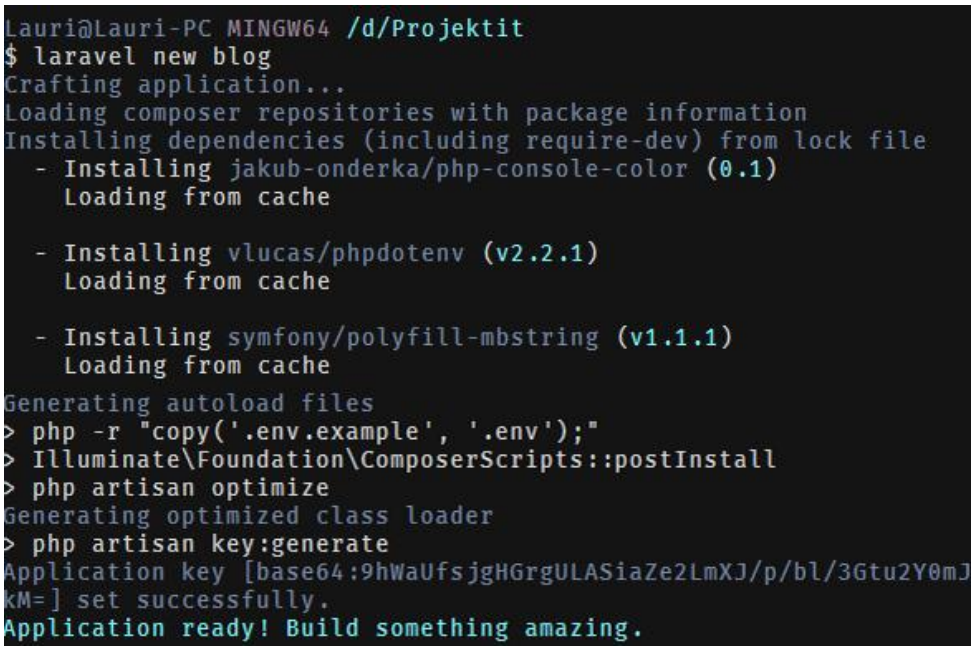
Composerin asentamisen jälkeen voidaan siirtyä asentamaan itse sivustolle tarpeellisia paketteja. Jotta Composer voisi asentaa Laravel projektille tarpeelliset paketit, suositellaan Laravel 5.2 asennusohjeissa käyttämään. Laravel/Installer Composer pakettia. Laravel Installerin asentaminen isäntäkoneelle onnistuu seuraavalla seuraavasti (Otwell, 2016):

```
Lauri@Lauri-PC MINGW64 /d/Projektit
$ composer global require laravel/installer
Changed current directory to E:/Users/Lauri/AppData/Roaming/Composer
Using version ^1.3 for laravel/installer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing laravel/installer (v1.3.3)
    Downloading: 100%

Generating autoload files
```

Kuvio 6: Komento Laravel Installerin asentamista varten

Laravel Installerin asentamisen jälkeen voidaan isäntäkoneella suorittaa komento, joka asentaa uuden Laravel projektin tarvitsemat PHP paketit. Komento tulisi suorittaa siinä kansiossa, johon haluat sijoittaa uuden projektin tiedostot sisältävän kansion.



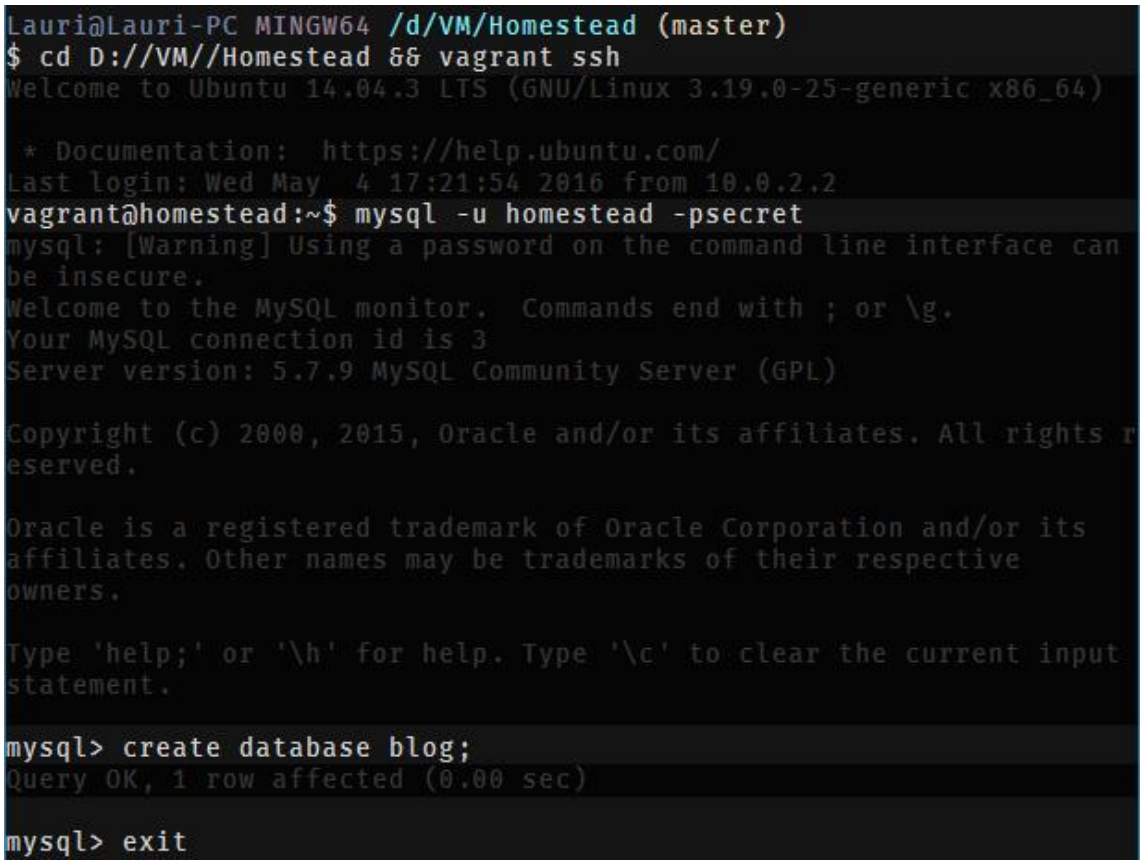
Kuvio 7: Composer komento Laravelin asentamista varten

Komento asentaa uuden Laravel projektin komennossa määritettyyn kansioon (tässä tapauksessa blog). Jos kehitysympäristö on käynnistetty ja asennettu oikein, sekä isäntä tietokoneen hosts-tiedosto on määritelty vastaamaan kehitysympäristön asetuksia, tulisi uuden projektin nyt löytyä esim. osoitteesta: www.blog.dev.

4.2 Tietokanta

Tässä projektissa käytettiin tietokantana MariaDB:ä, joka tulee myös esiasennettuna Laravel Homestead-virtuaalikoneeseen. MariaDB sivuston mukaan (2016) MariaDB on yksi maailman suosituimmista tietokantapalvelimista maailmassa. Sen käyttäjiin kuuluvat mm. Wikipedia, Facebook ja Google.

Jotta projektissa käytettävä tietokanta voitaisiin luoda, täytyy tietokanta ensin luoda virtuaalikoneessa. Virtuaalikoneeseen ja SQL-tietokantaan yhdistäminen vaihtelee tapauskohtaisesti. Tässä osiossa käydään läpi ainoastaan tapa, jolla tietokanta voidaan luoda Laravel Homestead-virtuaalikoneeseen. Koska kehityksessä käytetyssä tietokoneessa Laravel Homestead on asennettu kansioon: D:\VM\Homestead, suoritetaan komentorivillä seuraavat komennot:



Kuvio 8: Tietokannan asentaminen Laravel Homesteadissa

Nyt virtuaalikoneesta löytyy blog-niminen tietokanta, jota voidaan jatkossa käyttää tulevan sivuston tietojen tallentamiseen. Asennettu Laravel sovellus ei kuitenkaan vielä tiedä luodusta tietokannasta, joten avataan asennetun blog sovelluksen juuri kansiosta löytyvä ympäristökohtainen .env määritystiedosto ja muokataan DB_DATABASE-kohta vastaamaan juuri luotua tietokantaa: DB_DATABASE=blog

Laravel sisältää useita kehittämistä helpottavia komentoriviltä ajettavia komentoja. Palvelua kutsutaan nimellä artisan. Yksi artisanin komennoista on php artisan make, jonka avulla voidaan luoda erilaisia Laravel-sovellusten tarvitsemia ominaisuuksia. php artisan make-komento luo lähes poikkeuksetta sovelluksen kansiorakenteen sopivaan paikkaan tiedoston, joka sisältää koodi rungon pyydetystä määreestä.

Yksi edellä mainitun komennon määreistä on make:migration. Tällä komennolla voidaan luoda tiedosto, joka helpottaa huomattavasti tietokannan taulujen luomista ja niiden versiointia versionhallintaa varten. Migration PHP luokkatiedostoissa määritetään tietokantaan luotavan taulun rivit ja rivien mahdolliset yhteydet.

Taulun nimi	Migration komento
pages	php artisan make:migration create_pages_table –create
gallery_images	php artisan make:migration create_gallery_images_table –create
tags	php artisan make:migration create_tags_table –create
gallery_image_tag	php artisan make:migration create_ gallery_images_tags_table –create

Taulukko 2: Luotavat migration PHP luokkatiedostot

Edellä mainittujen migration PHP luokkien luomisen jälkeen, täytetään luokat suunnitteluvaiheessa luotujen taulujen mukaan. Make:migration --create-komento esiasentaa PHP luokkiin metodit up(), johon luodaan automaattisesti rivit kohdille id ja timesteamps, ja down(), jonka avulla voidaan tarvittaessa pudottaa taulu kutsuttaessa komentoriviltä komentoa php artisan migrate:rollback.

```
public function up()
{
    Schema::create('pages', function (Blueprint $table) {
        $table->increments('id');
        $table->string('title');
        $table->string('slug');
        $table->text('body');
        $table->timestamps();
    });
}
```

Kuvio 9: Pages taulun up-metodi

Tarvittavien taulujen up-metodien luomisen jälkeen voidaan komentoriviltä ajaa komento php artisan migrate. Tämä komento luo tietokantaan taulut, jotka perustuvat aikaisemmin

luotuihin migrate PHP luokkiin. Migrate komennon jälkeen on tietokanta valmis käyttöä varten.

Nimi	Rivit	Koko	Kone	Kommentti	Tyyppi
gallery_images	0	16,0 KiB	InnoDB		Table
gallery_image_tag	0	48,0 KiB	InnoDB		Table
migrations	6	16,0 KiB	InnoDB		Table
pages	0	16,0 KiB	InnoDB		Table
password_resets	0	16,0 KiB	InnoDB		Table
tags	0	16,0 KiB	InnoDB		Table
users	0	16,0 KiB	InnoDB		Table

Kuvio 10: Valmis blog-tietokanta-taulu (HeidiSQL)

4.3 MVC-mallin mukainen sovelluksen toteuttaminen

Nyt kun tietokanta on valmiina vastaanottamaan sovelluksen lähettämiä tietoa, voidaan aloittaa itse sovelluksen kehittäminen. Laravel tarjoaa kaikille ohjelmistokehystä käyttäville kehittäjilleen komennon, jonka avulla voidaan luoda yksinkertainen kirjautumisjärjestelmä. Komennolla php artisan make:auth voidaan luoda kirjautumisjärjestelmän perusta. Komento luo kirjautumisjärjestelmää varten tarpeelliset näkymät, kontrollerit ja päivittää sovelluksen polut vastaamaan uutta konfiguraatiota.

Kirjautumisjärjestelmä on hyvä luoda aikaisessa vaiheessa. Näin voidaan jakaa jatkossa luotavat sovelluksen reitit sen mukaan, halutaanko käyttäjän olevan kirjautuneena (esim. sivujenluominen, muokkaaminen ja poistaminen), vai saavatko kaikki vierailijat nähdä reitistä löytyvän sivun sisällön (esim. sivujen näyttäminen).

Kuvio 11: Sivuston etusivu make:auth-komennon jälkeen

4.3.1 Polut

Sovelluksen polut (route) määrittävät ne sivut, jotka sovelluksen halutaan tunnistavan. Jos käyttäjä päätyy muille, kuin polkujen määrittämille sivuille, palautetaan hänelle ns. 404-sivu, jossa kerrotaan käyttäjän päätyneen sivulle, jota ei sovelluksesta löydy. Polut ovat sovelluksen selkäranka, jotka ovat mukana lähes kaikissa sovelluksen toiminnoissa.

Laravel tarjoaa hyvin yksinkertaisten sovellusten kehittäjille vaihtoehdon route:resource, joka helpottaa polkujen määrittämistä. Määrittämällä polun käyttämään resource attribuuttia, voidaan yhdellä koodirivillä määrittää kaikki tarpeelliset REST-pohjaiset päätepisteet (endpoint). Resource-tyyppiset polut käyttävät tietokanta-taulun ID-saraketta hakeakseen mallin tietokannasta. Tässä opinnäytetyössä ei valitettavasti voida tätä ominaisuutta käyttää, koska haluamme sivujen löytyvän esimerkiksi osoitteesta blog.dev/sivu, eikä blog.dev/1, jossa sivu esittää Page-mallin slug-solua ja 1 esittää Page-mallin ID-solua.

REST-pohjaiset polut voivat Laravelissa sisältää seuraavia polkuja: get, post, put, patch, delete, match, any ja options. Polut määritetään oletusarvoisesti app/Http/routes.php tiedostossa. Sovelluksen poluissa näkyy myös as – uses attribuutti yhdistelmä, jonka avulla polkua voidaan helposti kutsua muualla sovelluksen koodissa. {slug} parametriä kutsuva viimeinen polku on sijoitettu tiedoston alalaitaan, jotta muut polut haetaan ennen yleistä polkua. Käytännössä tämä siis tarkoittaa sitä, että jos sivu ei kuulu mihinkään yllä olevista poluista, käytetään viimeistä yleistä polkua.

```php
<?php

Route::group(['middleware' => 'web'], function () {
    Route::auth();

    Route::get('/', 'PagesController@etusivu');

    Route::get('tietoa', 'PagesController@tietoa');
    Route::get('etusivu', function() { return redirect('/'); });
    Route::get('galleria', 'PagesController@galleria');
    Route::get('ehdotelmia', 'PagesController@ehdotelmia');
    Route::get('yhteystiedot', 'PagesController@yhteys');

    Route::post('yhteys-send', ['as' => 'yhteys_send', 'uses' =>
        'ContactController@send']);
```

Kuvio 12: Julkiset polut

4.3.2 Kontrollerit

Sovelluksen kontrollerit välittävät tietoa malleilta näkymille. Kontrollereilla voidaan myös käsitellä ja validoida tietoa ennen sen tietokantaan tallentamista. Näin voidaan varmistua siitä, että syötetty tieto on sovelluksen tietokantamalliin sopivaa. Kontrollerien kautta tieto siirtyy kaikille sovelluksen osa-alueille ja näin on myös mukana suuressa osassa sovelluksen toimintaa.

Laravel tarjoaa kehittäjille jälleen kätevän make:controller-komennon, jonka avulla voidaan luoda MVC-mallin mukaisia kontrollereita. make:controller-komento luo app/Http/controllers-kansioon PHP-luokan, joka mukailee nimeltään asetettavaa luokan nimeä, esim php artisan make:controller PagesController. Esimerkkinä PagesController-luokassa määritetään niiden julkisesti näkyvissä olevien sivujen metodit, jotka halutaan sivustolla näyttää.

Yleensä REST-tyyppisissä kontrollereissa voidaan määrittää seuraavat toiminnot: show, create, store, edit, update ja delete. Kontrollereille voidaan tietysti lisätä niin paljon toimintoja ja metodeita, kuin halutaan. Laravel antaa hyvät mahdollisuudet kuitenkin erottaa kontrollerille kuulumattomia toimia muille sovelluksen osa-alueille, esim. pyyntöjen oikeuksien tarkastaminen (middleware), lomakkeiden pyyntöjen validointi (form request validation), mallien tapahtumien hallinta (model events) jne.

```php
<?php

namespace App\Http\Controllers;

use DB;
use App\Page;
use App\GalleryImage;
use App\Http\Requests;
use Illuminate\Http\Request;

class PagesController extends Controller
{
    public function etusivu()
    {
        $page = Page::where('slug', 'etusivu')->firstOrFail();

        return view('pages.etusivu', compact(['page', 'sliderImages']));
    }

    public function tietoa()
    { ...
    }

    public function ehdotelmia()
    { ...
    }

    public function galleria()
    { ...
    }

    public function yhteys()
    { ...
    }
}
```

Kuvio 13: PagesController PHP-luokka

4.3.3 Mallit

MVC arkkitehtuurissa mallit muokkaavat malli-objektien tallennettavaa tietoa ja välittävät sitä kontrollereille ja sitä kautta tietokantaan (Kuvio 1: MVC-arkkitehtuurin toiminta.) Nimensä mukaisesti mallit ovat kaiken tiedon pohjapiirustuksia siitä, mitä kaikkea tallennettaviin objekteihin halutaan sisällyttää. Laravelin malli-luokat perustuvat Eloquent ORM ActiveRecord toteutukseen. Eloquent ORM toteutuksessa esiintyy muutamia erityisiä muuttujia, joita syöttämällä voidaan muokata yksittäisten mallien toimintaa. Edellä mainittuihin muuttujiin kuuluvat mm. mallin tietokannasta löytyvän taulun nimen muuttaminen (jos ei vastaa normaalia käytäntöä, protected $table), mallin täytettäväksi hyväksyttävät tietokannan rivit (protected $fillable) ja kyseisen mallin suhteet muihin sovelluksen malleihin.

Laravel tarjoaa jälleen hienon tavan luoda uusia malli luokkia komentorivin välityksellä: php artisan make:model modelName (Otwell, 2016), jossa modelName on muuttuja, jonka avulla voidaan määrittää luotavan mallin nimi (esim. Page.) Mallien nimeämisessä tulee huomioida muutama seikka: Mallin nimen tulisi alkaa isolla alkukirjaimella ja nimi tulisi kirjoittaa CamelCase tyylillä, esim. GalleryImage. Käytettäessa mallin CamelCase-nimeä, oletetaan tietokannasta löytyvän vastaava monikossa kirjoitettava snake_case niminen taulu, esim. gallery_images. Kuten aikaisemmin tässä luvussa mainittiin, nämä ominaisuudet voidaan kuitenkin muuttaa käyttämällä mallin protected $table muuttujaa.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Page extends Model
{
    protected $fillable = [
        'title',
        'image',
        'thumbnail',
        'body'
    ];
}
```

Kuvio 14: Page mallin PHP-luokka

4.3.4 Näkymät

MVC arkkitehtuurin mukaiset näkymät esittävät kontrollerien niille tuomaa tietoa käyttäjän päätteessä, usein HTML ja Javascript muodossa. Näkymät ovat käyttäjien ikkuna sovelluksen toiminnollisuuteen ja ovat näin hyvin tärkeässä roolissa sivustoa luotaessa. Laravelissa kontrollereilta tuodut muuttujat saadaan näkymissä näkymään Laravelin Blade:n avulla. Laravel toimitetaan myös Laravel Elixir Gulp kääntäjän kanssa, jonka avulla voidaan helpottaa käyttöliittymän suunnitteluun liittyviä tehtäviä. Näitä tehtäviä ovat mm. tyylitiedostojen yhdistäminen yhteen tiedostoon, Javascript koodin yhdistäminen ja optimoiminen, sekä uusien Javascript ominaisuuksien tukeminen erinäisten kääntäjien avulla.

Laravelissa näkymät sijaitsevat resources/views-kansiossa, josta ne automaattisesti haetaan kutsuttaessa. Laravel Blade mahdollistaa HTML pohjien (layout) käyttämisen, jossa pää-pohjassa (master layout) määritetään miltä sivuston pitäisi suurimmilta osin näyttää ja sitten tulostetaan sisällölle sopivaan kohtaan yksittäisten sivujen sisältö. Laravel Blade myös

mahdollistaa PHP koodin suorittamisen pääosin HTML-koodia sisältävien tiedostojen sisällä. Tällä tavoin kontrollerilta tuotavia muuttujia voidaan käyttää myös näkymissä.

Valitettavasti sovelluksia luotaessa näkymiä luodaan niin monia, että tässä opinnäytetyössä ei voida kaikkia tämän sovelluksen näkymiä esittää. Jotta lukija voisi kuitenkin toistaa opinnäytetyön sisällön, tarjotaan opinnäytetyössä tärkeimmät näkymät toiminnollisuuden kannalta.

Alustavasti kokeiltiin hyvin yksinkertaisella pohjalla ja etusivu-sivulla että kaikki tähän mennessä asennettu toimii odotetulla tavalla.

```
<!DOCTYPE html>
<html lang="fi">
<head>
    <meta charset="UTF-8">
    <title>Blog</title>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>

    @yield('content')

</body>

</html>
```

Kuvio 15: Näkymä - partials/app.blade.php

```
@extends('layouts.app')

@section('content')
    <p>Etusivu</p>
@stop
```

Kuvio 16: Näkymä: pages/etusivu.blade.php

Kuvio 17: Etusivu alustavan kokeilun jälkeen

Toiminnan varmistamisen jälkeen, voidaan muokata /partials/app.blade.php näyttämään halutut sovelluksen osat. Pää-pohja (app.blade.php) toimii käytännössä niin, että @include() direktiiveillä voidaan määrittää joka kerta sivulla näytettävät osiot ja @yield() direktiiveillä taas tulostetaan näkymien yksilölliset tiedot.

```
<!DOCTYPE html>
<html lang="fi">
<head>
    <meta charset="UTF-8">
    <title>Blog</title>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="{{ elixir('css/app.css') }}">
</head>
<body>

    <div class="wrapper">

        @include('partials.nav')

        @include('partials.flash')

            @yield('content')

        </div> <!-- .content-wrapper -->

    </div> <!-- .wrapper -->

    @include('partials.footer')

</body>

@yield('additional.scripts')
<script src="{{ elixir('js/all.js') }}"></script>

</html>
```

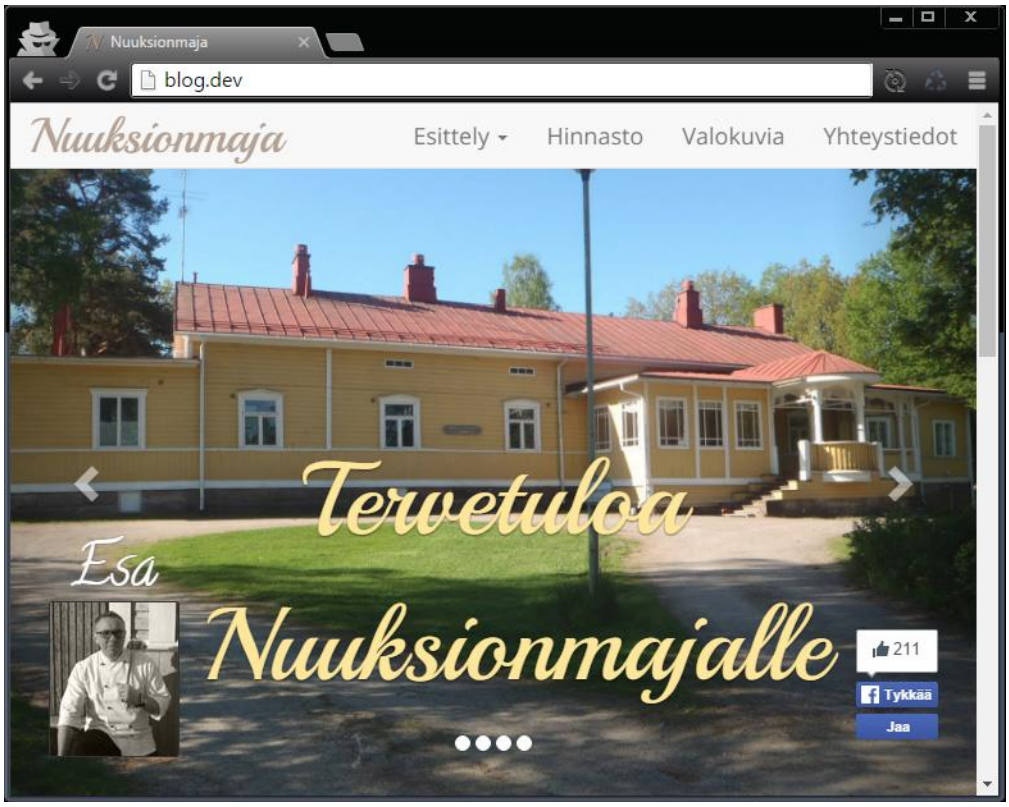
Kuvio 18: Uudistettu partials/nav.blade.php

Tässä vaiheessa voidaan myös lisätä sivuille tarpeelliset Bootstrap käyttöliittymä kehyksen luokat, joilla saadaan sivuun selkeämpi rakenne. Sisällyttämällä Bootstrapin CSS-tiedostot päänäkymään (app.blade.php), saadaan kaikki Bootstrap kehyksen CSS-luokat sivuston käyttöön. Tässä projektissa käytettiin Laravel Elixiriä yhdistämään kirjastojen ja projektin omat tiedostot yhdeksi tiedostoksi, jolla voidaan vähentää HTTP pyyntöjen määrää ja näin

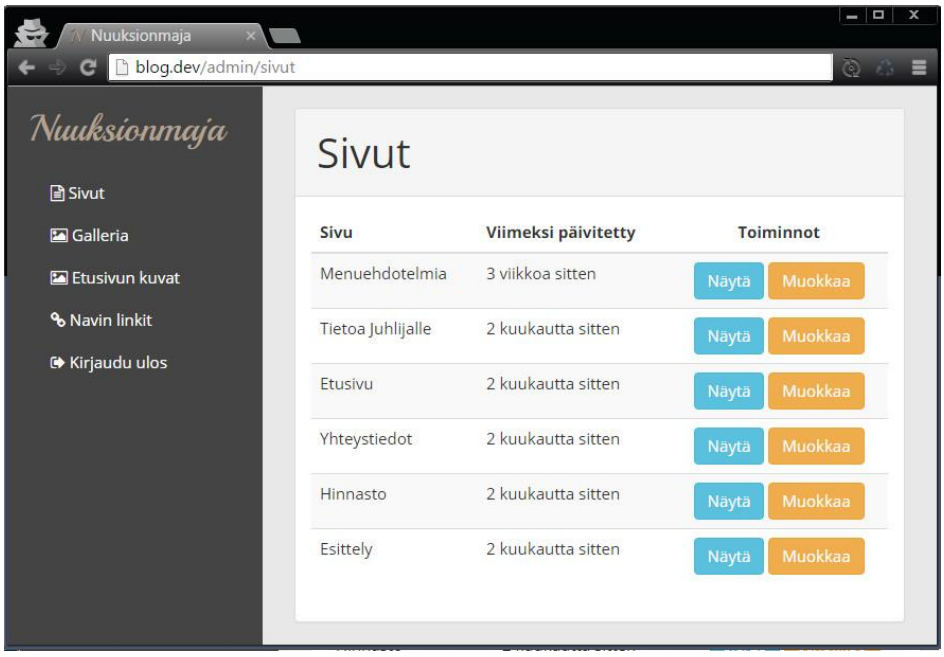
myös lyhentää sivujen latausaikaa.

Näkymän pages/galleria.blade.php alaosassa näkyvä additional.scripts-kohta tulostaa sivulle käytettävien javascript kirjastojen vaihtoehtoiset asetukset. Myös app.blade.php:n alaosasta löytyvä additional-scipts varmistaa sen, että kirjastojen asetukset ladataan vasta sen jälkeen, kun itse pääkirjasto on ladattu. Tässä tapauksessa galleria-sivulla halutaan hidastaa Lightbox-kirjaston kuvien muuntautumisnopeutta ja poistaa albumeista nimikkeet.
Näkymien lisäämisen, Bootstrap-kirjaston, projektikohtaisten CSS-määritysten ja Etusivu-sivun tietokantaan lisäämisen jälkeen etusivu on valmiina ottamaan vastaan käyttäjän syöttämää tietoa.

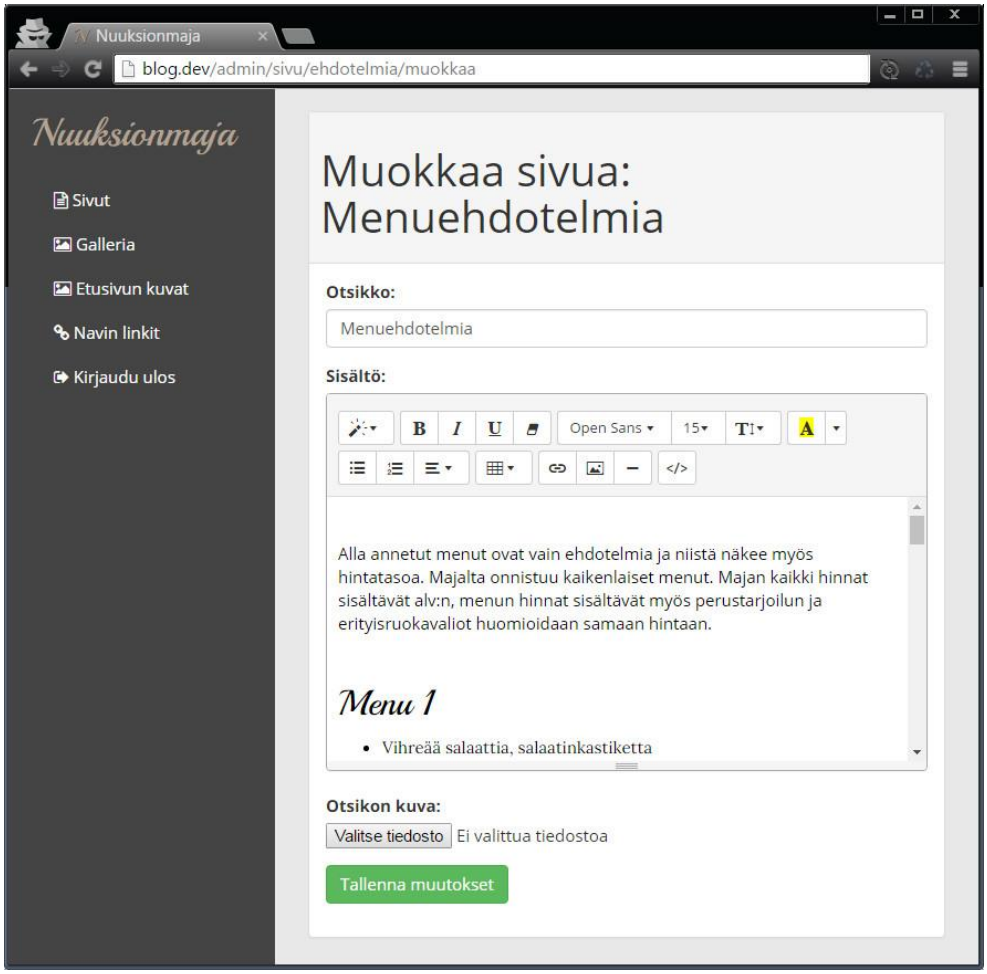
Kuvio 19: Päivitetty etusivu



Kuvio 20: Uudelleen tyylitelty etusivu



Kuvio 21: admin/sivut ylläpitosivu



Kuvio 22: admin/sivu/ehdotelmia/muokkaa sivu

5 Yhteenveto

Tämän opinnäytetyön tavoitteena oli luoda Nuuksionmajan juhlatilalle helppokäyttöinen, informatiivinen ja toimiva verkkosivusto, joka mahdollistaisi myös ammattilaisesta riippumattoman sisällön ylläpidon. Ulkoasun ja sivustolta löytyvän informaation toivottiin pohjautuvan Nuuksionmajan aikaisempaan verkkosivustoon. Luotavan sivuston toivottiin myös olevan responsiivinen. Responsiivisuutta toivottiin, koska toimeksiantaja tulisi päivittämään sivustoa pääasiassa tabletilla ja suuri osa sivustolla vierailevista Nuuksionmajan asiakkaista käyttävät mobiililaitteita.

Opinnäytetyön tuloksena Nuuksionmajalle valmistui tavoitteiden mukainen verkkosivusto, jossa toimeksiantaja pystyy itse päivittämään sivuston sisältöä. Verkkosivusto toteutettiin käyttämällä moderneja web-kehityksen työkaluja ja toimintatapoja. PHP-pohjainen Laravel ohjelmistokehys toimi kehitettävän sivuston alustana ja ulkoasu pohjautuu Bootstrap front-end-kehykseen. Laraveliin päädyttiin sen laajan dokumentaation ja suosion perusteella.

Bootstrap taas oli kehittäjälle ennestään tuttu responsiivinen front-end-kehys, jonka avulla sivuston sisältö pystyttiin esittämään responsiivisena.

Lähteet

Painetut lähteet

Dangar, H. 2013. Learning Laravel 4 Application Development. Packt Publishing Ltd

Saunier, R. 2014. Getting Started with Laravel 4. Packt Publishing Ltd

Somasundaram, R. 2013. Git: Version Control for Everyone. Packt Publishing Ltd

Julkaisemattomat lähteet

Airaksinen, T. 2009. Toiminnallinen opinnäytetyö tekstinä. Viitattu 23.5.2016. http://www.slideshare.net/TiinaMarjatta/toiminnallinen-opinnytety-tekstin

Composer.org. 2016. Download Composer, Windows Installer. Viitattu 15.5.2016. https://getcomposer.org/download/

dragonfire1119. 2015. How to use jquery ui sortable in laravel. TutsGlobal.com

MariaDB Corporation Ab. 2016. About MariaDB. Viitattu 22.5.2016. https://mariadb.org/about/

Otwell, T. 2016. Installing Laravel. Viitattu 15.5.2016. https://laravel.com/docs/5.2/installation

Otwell, T. 2016. Defining Models. Viitattu 10.4.2016. https://laravel.com/docs/master/eloquent#defining-models

Summernote Team. 2016. Deep dive. Viitattu 15.5.2016. http://summernote.org/

Way, J. 2016. Laravel From Scratch. Laracasts Inc. Chattanooga, TN.

Way, J. 2015. Build "ProjectFlyer" With Me. Laracasts Inc. Chattanooga, TN.

Kuviot

Taulukot

Liitteet