Oona-järjestelmän arkkitehtuuridokumentti

8. toukokuuta 2015

Sisältö

Kuvat

Sovellus on tehty php:lla käyttäen Laravel-sovelluskehystä (versio 4.2). Sovellus on tämän ansiosta pitkälti MVC-mallin mukainen. Lisäksi osa sovelluksen käyttöliittymän toiminnallisuudesta on toteutettu Javascriptilla käyttäen JQuery-kirjastoa. Projektissa käytetään composer-riippuvuuksienhallintaa. Projektin riippuvuudet on määritelty composer.json -tiedostossa. Jos composer on asennettu järjestelmään, voi riippuvuudet ladata ja päivittää suorittamalla projektin juuressa komennon composer update .

1 Laravel ja MVC-malli

Laravelin yleinen toiminta selviää Laravelin omasta dokumentaatiosta. Tässä on keskitytty esittelemään Oona-järjestelmälle ominaisia piirteitä.

MVC-mallissa mallit (models) ovat luokkia, jotka kuvaavat järjestelmään tallennettuja tietokohteita. Tämän järjestelmän model-luokkia ovat Answer (jatkossa nimikkeellä vastaus), Attachment (jatkossa nimikkeellä liite), Comment (jatkossa nimikkeellä kommentti), Committee (jatkossa nimikkeellä toimikunta), Lurker, Poll (jatkossa nimikkeellä kysely), Timeidea ja User (jatkossa nimikkeellä käyttäjä). Modelit löytyvät hakemistosta /app/models .

Tarkoituksena on, että mallit tarjoavat kaikki tarpeelliset SQL-kyselyt, vaikka tätä periaatetta rikotaan hieman esimerkiksi tilastojen luomisessa. Lisäksi Laravelin tarjoama syntaksi tekee kyselemisestä niin helppoa, että joissakin tapauksissa querybuilderilla rakennettuja kyselyitä löytyy myös controllereista. Raakaa SQL:ää on vältetty, koska se aiheuttaa vaikeuksia eri tietokantamoottoreihin porttaamisessa. Kuitenkin tilastoista vastaava controller rikkoo myös tätä periaatetta.

Tietokohteiden välisiä yhteyksiä on mahdollista kysellä malleilta käyttämällä Eloquent ORM:n tarjoamia valmiita välineitä (tämän kirjoittajaa eniten auttanut tutoriaali: https://scotch.io/tutorials/a-guide-to-using-eloquent-orm-in-laravel myös Laravelin dokumentaatio on hyvä).

Mallien väliset suhteet ja merkitys tietosisällön kuvaamisen kannalta on kuvattu tietokantakaaviossa. Muutamia kohtia on kuitenkin paikallaan selventää. Järjestelmän doodle-tyyppinen kyselyominaisuus on toteutettu siten, että kyselyihin liittyy ajankohtia (esim. "ti 12-14") ja näihin vastauksia. Lurker-oliot ovat järjestelmän ulkopuolisia vastaajia, joista kukin liittyy tiettyyn kyselyyn. Kuhunkin toimikuntaan liittyy liitteitä. Liitteiden lukemisesta pidetään kirjaa monesta moneen yhteyttä kuvaavassa tietokantataulussa.

2 Kontrollerit

Kontrollerit (controllers) käsittelevät käyttäjän pyyntöjä ja muuttavat modeleita ja näkymiä käskyjen mukaisesti. Kontrollereilla on useita funktioita, joilla pyyntöjä käsitellään. Laravel tarjoaa yksinkertaisen tavan generoida CRUD-controller-pohjia. Kaikissa tapauksissa kuitenkaan koko CRUD-toiminallisuutta ei ole toteutettu, jolloin controlleriin on jäänyt automaattisen generoinnin jäljiltä tyhjiä funktioita. Kontrollerit löytyvät /app/controllers -hakemistosta. Järjestelmässä on seuraavat kontrollerit:

AttachmentController Controller hallitsee toimikuntaan liittyviä liitetiedostoja. Julkisessa rajapinnassa on kolme funktiota: download, store ja destroy. Destroy ja store ovat vain adminin, download kaikkien toimikunnan jäsenten

käytettävissä. Oikeuksien ja tiedoston nimen tarkistamiseen tarvitaan jonkin verran toimenpiteitä. Mikäli tiedoston nimi on jo käytössä, nimen perään lisätään järjestysnumero suluissa ennen tallentamista. Tämä on toteutettu rekursiolla, jonka oletuksen mukainen syvyys PHP:ssa on sata toistoa.

CommentController Controller hallitsee kommentteja, joita voi olla sekä toimikunnissa että kyselyissä. Controllerissa on ainoastaan store-funktio, sillä kommentteja ei voi muokata tai poistaa. Controller toteutettiin ensin vain kyselyitä varten, ja toimikuntien kommentointi ominaisuus lisättiin jälkikäteen. Toimikuntiin voivat lisätä kommentteja ainoastaan toimikunnan jäsenet ja admin, kun taas kyselyissä kuka tahansa voi kommentoida. Tämän vuoksi store-funktio on sisältää kaksi eri toiminnallisuutta if-lauseissa.

AnswerController Controller hallitsee kyselyihin annettuja vastauksia. UpdateSopivuus-funktiota käytetään, kun käyttäjä on valinnut vastauksensa ja painanut tallenna vastaukset-nappia kysely näkymässä. Vastaukset välitetään funktiolle html-formin kautta taulukkona. CreateAnswer-funktio luo uuden vastauksen.

CommitteeController Controller vastaa toimikuntiin liittyvistä toiminnoista, jotka toteuttavat CRUD-nelikon.

LurkerController Controller hallinnoi kyselyiden vastaajia, jotka eivät ole kirjautuneita käyttäjiä. Store-funktio luo uuden lurkerin ja alustaa tämän vastaukset 'eivastattu'-arvoiksi. Destroy-funktio poistaa lurkerin ja tämän kaikki vastaukset.

PollController Controller vastaa kyselyiden hallinnoinnista. Se toteuttaa CRUD-nelikon.

StatsController Controller vastaa "Tilastoja"-näkymässä näytettävien tietojen tuottamisesta. Index-funktio on vaikealukuinen. Siinä luodaan taulukko, johon kootaan erilaisia summa tietoja sql-kyselyillä.

TimeideaController Controller vastaa kyselyissä käytettävistä ajankohdista.

LoginController Controller hallitsee kirjautumista. Toiminallisuus on suoraviivainen laravelin tarjoaman Auth-olion avulla.

UserController Controller vastaa kaikista rekisteröityneisiin käyttäjiin liittyvistä toiminnallisuuksista. Se toteuttaa CRUD-nelikon.

3 Näkymät

Näkymät (views) ovat käyttäjälle renderöitäviä html-sivuja. Näkymiä on tietokohteiden listaamista ja muokkaamista varten. Näkymät löytyvät /app/views -hakemistosta. Näkymät on ryhmitelty tiettyyn resurssiin liittyviin toiminallisuuksiin. Esimerkiksi on olemassa views/user-kansio, jossa on näkymät käyttäjien listaukseen tai yhden käyttäjän tietojen näyttämiseen. Niinpä näkymien tarjoamia palveluita voi yhdistellä, esimerkiksi käyttäjän sivulla näytetään listoja kyselyistä.

Näkymien koodit ja toiminnallisuudet ovat pääosin hyvin suoraviivaisia. Yksi mahdollisesti hämäävä piirre liittyy näkymien ja kontrollerien kommunikaatioon. Näkymien koodissa oletetaan, että käytettävissä on näytettäviä olioita kuvaava

muuttuja, esim. users, jonka controller tarjoaa. Muutamassa kohdassa näytetään saman näkymän tarjoama listaus, mutta eri tietueilla. Esimerkiksi avoimet ja suljetut kyselyt käyttävät molemmat kysely-näkymän tarjoamaa listaus-toiminnallisuutta. Koska listaus toiminallisuus näyttää "polls"-muuttujassa olevat tietueet, tähän pääsemiseksi muttujan arvo vaihdetaan näkymässä sen jälkeen kun ensimmäinen lista on näytetty.

4 Tietokanta

Projektin yhtenä vaatimuksena oli, että järjestelmä rakennetaan käyttämään MySql-tietokantaa. Laravel tarjoaa yleistetyn alustan tietokantojen käytölle. Tästä syystä tietokannan käyttö oli vaivatonta. MySQL-tietokannan lisäksi ryhmä käytti PostgreSQL-tietokantaa Continuous deployment -palvelimellaan.

Tietokanta on abstrahoitu laravelin kautta ja toteutettu migraatioiden avulla. Migraatiot löytyvät hakemistosta /app/database/migrations .

Kuvaus sovelluksen tietokantatauluista löytyy tietokantakaaviosta, joka on kuvio 1. Ei tosin kannata uskoa kaikkea mitä tietokantakaavio kertoo esim. taulujen attribuuteista, ne näkee migraatioista.

Attachments Taulussa on toimikuntiin liittyvät liitetiedostot. Kustakin on tallennettu sen tiedostopolku (hämäävästi nimellä file), tiedostonimi sekä toimikunnan id. Tiedostot itsessään eivät siis ole tietokannassa, ainoastaan tiedostopolut on tallennettu tietokantaan.

Committees Taulu kuvaa toimikunnan tietoja. Toimikunnalla on järjestämisajankohta,

Polls Taulu kuvaa kyselyn tietoja.

Users Taulu kuvaa järjestelmän käyttäjän tietoja.

Participants Taulu kuvaa kyselyn osallistuja. Taulu oli alun perin välitaulu monesta-moneen-yhteyksiä kuvattaesa, mutta osallistujalla on myös itsenäistä tietosisältöä (nimittäin isselected-attribuutti, joka kertoo onko admin valinnun käyttäjän).

Answers Taulu kuvaa yhden käyttäjän antamaa vastausta tiettyyn ajankohtaehdotukseen.

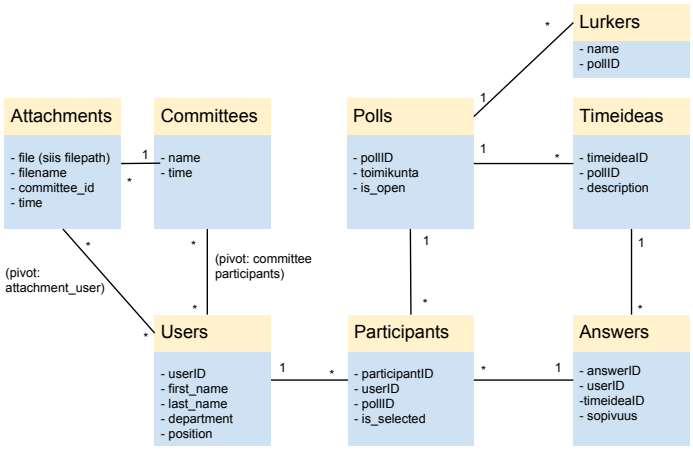
Timeideas Taulu kuvaa ajankohtaehdotusta. Tämä on siis esimerkiksi "tiistai 12-14"kyselyn rivinä.

Lurkers Taulu kuvaa kyselyn vastaajaa, joka ei kuitenkaan ole rekisteröitynyt käyttäjä järjestelmässä.

5 Reititys

Tiedostoon routes.php on määritelty kaikki mahdolliset polut sivustolla. Tiedosto vastaa myös osittain pääsynvalvonnasta. Käyttäjän kyselyt tarkistetaan ensin routes.php:ssa, josta kysely ohjataan oikean kontrollerin oikeaan funktioon käsiteltäväksi.

Kuva 1: Tietokantakaavio



6 Testaus

Automatisoituun testaukseen käytetään Codeception-testikehystä. Codeceptionilla ajetaan phpunit-yksikkötestit ja lisäksi funktionaaliset testit. Projektin aikana olimme tehneet acceptance-testejä, mutta teknisten ongelmien ja aikataulurajoitteiden takia jouduimme luopumaan niistä. Projekti sisältää kuitenkin shell-skriptin, jonka avulla testit on helppo ajaa ja josta olisi apua myös acceptance-testejä ajettaessa.

Acceptance-testien suorittamiseen liittyy se ongelma, että niiden ajon aikana on käynnissä oltava www-selain, joka tekee pyyntöjä palvelimelle. Otimme käyttöön phantomjs-nimisen kevyen webkit-pohjaisen selaimen, joka on mahdollista ajaa komentoriviltä.

Testit on helpointa ajaa Laravel Homestead -virtuaaliympäristön sisällä. Projektin juuresta löytyy run-tests.sh -niminen shelliskripti, jonka avulla testit voi helposti suorittaa. Skripti asentaa phantomjs-selaimen homesteadiin, käynnistää sen ja suorittaa testit, jonka jälkeen phantomjs sammutetaan.