jamk.fi

Vikatikettijärjestelmän toteutus ja käyttöönotto

Jussi Heikkinen

Opinnäytetyö
Lokakuu 2016
Tekniikan ja liikenteen ala
Insinööri (AMK), Ohjelmistotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Heikkinen Jussi	Opinnäytetyö, AMK	30.10.2016
	Sivumäärä 36	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Vikatikettijärjestelmän toteutus ja käyttöönotto

Tutkinto-ohjelma

Ohjelmistotekniikan koulutusohjelma

Työn ohjaaja(t)

Rantala, Ari

Toimeksiantaja(t)

NaturVention Oy

Tiivistelmä

NaturVention Oy valmistaa älykkäitä viherseiniä. Toimeksiantaja tarvitsi helppokäyttöistä työtehtävien ja vikatikettien hallintajärjestelmää osaksi heidän omaa intranetpalveluaan. Tikettijärjestelmä tuli pääasiallisesti huoltoryhmän käyttöön, mutta myös kasvi-, kehitys- ja myyntiosastot ottivat sen osaksi toimintaansa. Järjestelmää käytetään sellaisten työtehtävien hallintaan, jotka eivät kuulu huoltoryhmän normaaleihin huolto- tai työtehtäviin. Tällaisia työtehtäviä ovat ainakin erilaisten sensorien vaihtaminen sekä kasvitaudeista ja ötököistä ilmoittaminen. Muille käyttäjille tiketit ovat tavallisia työtehtäviä. Tikettien hyöty on siinä, että kaikki työtehtävät ovat yhdessä paikassa, ja ne tulevat kuitatuiksi valmistuessaan. Käyttäjät tekevät tikettejä havaitessaan ongelmia tuotteissa tai työvälineissä. Seinien automaatio tekee tikettejä havaitessaan puutteita asetuksissa tai jos jokin sensori ei anna lukemaa.

Järjestelmän pohjana käytettiin osTicket-tikettijärjestelmää. NaturVention Oy:n valmiiseen ratkaisuun rakennettiin rajapinta, jonka kautta voitiin luoda tikettejä osTickettiin. Rajapintaan ohjelmoitiin mahdollisuus tikettien lukemiselle, kommentoimiselle, muokkaamiselle sekä varaamiselle. Tavoitteena oli se, että tiketoinnin toiminta intranetin kautta olisi niin sujuvaa, ettei käyttäjän tarvitsisi käyttää osTickettiä. Toteutuksessa käytettiin pääosin PHP-ohjelmointikieltä. Tikettirajapinta rakennettiin osaksi Laravelilla ohjelmoitua intranettiä. Käyttöliittymät tehtiin Bootstrapilla ja näkymiin lisättiin dynaamisuutta JavaScriptillä.

Toteutus oli toimiva ja se otettiin päivittäiseen käyttöön. Tikettijärjestelmään saatiin lisättyä kaikki toimeksiantajan vaatimat ominaisuudet. Tikettijärjestelmästä on tullut pääosin positiivista palautetta käyttäjiltä, joten käyttäjäkokemus on onnistunut.

Avainsanat (asiasanat)

Vikatikettijärjestelmä, työnohjaus, Laravel, PHP, osTicket, HTML, JavaScript, Bootstrap

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Heikkinen Jussi	Bachelor's thesis	30.10.2016
		Language of publication: Finnish
	Number of pages 36	Permission for web publication: x

Title of publication

Development of support ticket system

Degree programme

Software Engineering

Supervisor(s)

Rantala, Ari

Assigned by

NaturVention Oy

Abstract

NaturVention Oy is a company that creates intelligent green walls. They assigned a task to create a support ticket system to be accompanied with their intranet. The ticketing system is supposed to keep track of unusual tasks for their maintenance team. For example, a ticket should be done if a product has a fault that does not belong to the normal maintenance. Also, plant, research and development and sales teams use ticketing. They use it like a reminder carry out a specific task. Tickets are done by users and automation. Automation creates a ticket if it can not read a sensor or if there is a great change in climate around the wall. This gives the employees an advantage to react to those changes and to save the plants in a green wall.

The ticketing system was built on osTicket support ticketing system. The task was to implement ticketing to the employees' intranet Remo, thus osTicket was used as a storage for tickets. An interface was built in Remo to create all the functionality within the intranet. This interface can create a ticket to osTicket through its API, and it has all the functionality that a user would have in osTicket. Users can assign, comment, modify and close tickets in Remo. The ticketing interface was built on Laravel framework with PHP. The front end user interface was built using Bootstrap and JavaScript.

Ticketing system was a successful project. It has all the functionality that the assigner expected it to have. It works flawlessly, and it has received good feedback from the end users. Ticketing has made its way as a daily tool in NaturVention Oy. It has been mainly used by the maintenance team; however, due to its versatility the latest team to use it has been the sales team that gets a ticket when a wall is installed or moved to another space at a customer's premises.

Keywords/tags

Support ticket system, helpdesk, Laravel, PHP, osTicket, Bootstrap, JavaScript, HTML

Miscellaneous

Sisältö

Kuviot

Käsitteet

API

Application programming interface on ohjelmointirajapinta, jota voidaan kutsua ohjelman ulkopuolelta tai sisältä.

Bootstrap

Bootstrap on HTML:ää, CSS:ää ja JavaScriptiä yhdistävä ohjelmistokehys.

IDE

IDE eli integrated development enviroment on ohjelma, joka tarjoaa ohjelmistokehitykseen tarvittavat työkalut.

JSON

JavaScript-objektia vastaava tietorakenne, jolla on laaja tuki ohjelmointikielissä.

Laravel

PHP- ohjelmistokehys, joka tarjoaa MVC-mallin sekä laajan valikoiman työkaluja palvelinpuolen ohjelmointiin.

MariaDB

MySQL:stä otettu haara, joka on ohjelmistoyhteisön ylläpitämä.

MVC

Ohjelmoinnissa käytettävä ohjelmointitapa, jossa data, datan käsittely sekä esittäminen on jaettu omiin kerroksiin.

MySQL

Oraclen omistama suosituin avoimeen lähdekoodin perustuva relaatiotietokanta.

NGINX

NGINX on avoimeen lähdekoodiin perustuva http-palvelin, joka on tunnettu erittäin hyvästä suorituskyvystään staattisen sisällön palvelemisessa.

ORM

Tietokannan tauluista luotu malli ja rajapinta, jonka kautta dataa voidaan tallentaa ja tulostaa helposti.

osTicket

Suosituin avoimeen lähdekoodiin perustuva vikatiketti- ja asiakastukijärjestelmä.

PHP Hypertext Preprocessor

PHP on palvelinpuolen ohjelmointikieli, jonka avulla voidaan tehdä dynaamisia verkkosivuja.

POST

Http- protokollan metodi, jota käytetään, kun halutaan lähettää dataa palvelimelle, tai jos halutaan piilottaa parametrit http- pyynnön otsikkokenttään.

REST

Representational state transfer rajapinta eli rajapinta, joka vastaa, kun sitä kutsutaan oikeilla parametreilla.

VOC

Volatile organic compound eli VOC- yhdisteiden päästölähteitä ovat erityisesti rakennus- ja sisustusmateriaalit sekä pesuaineet. Etenkin VOC- yhdisteiden yhteisvaikutuksen epäillään aiheuttavan terveyshaittaa.

1 Työn lähtökohdat

1.1 Tausta

Työn toimeksiantaja NaturVention Oy valmistaa älykkäitä viherseiniä, jotka tarvitsevat kuukausittaista huoltoa. Huoltoa tekee tällä hetkellä parinkymmenen henkilön ryhmä. Seinän huoltokäyntiin voi sisältyä esimerkiksi kasvien vaihtamista tai seinän ulkoisen olemuksen kohentamista. Huoltokäynneillä tehdään myös tavallisista huoltokäynneistä poikkeavia tehtäviä, kuten ympäristön mittaukseen tarkoitettujen sensorien vaihtoja. Lisäksi seinien tekoäly nostaa hälytyksiä poikkeavista tilanteista tai vioista, jotka täytyy tarkistaa huoltokäynnillä tai etähallintajärjestelmästä. Koska yrityksen kasvuvauhti on ollut kova, työtehtävien hallintaan täytyi kehittää järjestelmä, jolla tehtävien hallittavuus sujuu järjestelmällisesti.

1.2 Tavoitteet

Tavoitteena oli suunnitella ja toteuttaa järjestelmä, joka toimii pääasiassa huollon ja tuotannon työkaluna. Järjestelmään voi lisätä työtehtäviä, tuotteessa havaittuja vikoja tai puutteita sekä muistutuksia. Seinissä toimiva tekoäly havainnoi seinän tilaa ja tekee vikatiketin, jos anturitietoja ei saada, asetuksia puuttuu tai jos ympäristön tekijät voivat aiheuttaa vaaraa viherkasveille. Järjestelmän täytyy toimia sulautettuna osana yrityksen omaa intranettiä Remoa, mutta sen täytyy toimia myös itsenäisenä järjestelmänä, sillä Remon käyttöoikeus on käyttäjien osalta rajattu osalta käyttäjistä. Tikettejä täytyy pystyä tekemään ainakin viherseinille, seinäryhmille, käyttäjille sekä käyttäjäryhmille. Tarkempi vaatimusmäärittely esitetään luvussa 6.

1.3 Toimeksiantaja

NaturVention Oy on suomalainen startup-yritys, jonka tavoitteena on edistää ihmisten hyvinvointia yhdistämällä teknologia ja viherkasvit. NaturVention Oy on perustettu vuonna 2011, ja se työllistää 32 henkilöä. Yrityksen kasvuvauhti on ollut kova, sillä se on keskimäärin kaksinkertaistanut henkilöstönsä vuosittain (Rekisteritiedot, NaturVention Oy 2016.)

NaturVention Oy:n viherseinässä eli Naavassa on hyvinä ilmanpuhdistajina tunnettuja huonekasveja, jotka puhdistavat ilmasta erilaisia haitallisia haihtuvia orgaanisia yhdisteitä eli VOC-päästöjä. Näitä päästöjä ovat muun muassa aromaattiset hiilivedyt, tolueeni, bentseeni ja aldehydit. Seinässä kasvit ovat ruukuissa, joissa on epäorgaaninen kasvualusta, jonka läpi huoneilmaa kierrätetään. Kasvien juurissa elävät mikrobit käyttävät ilman epäpuhtauksia ravintonaan ja siten hävittävät niitä huoneilmasta. Idea on saanut lähtönsä Nasan tutkimuksista, joissa selvitettiin keinoja ilman puhdistamiseen avaruudessa. Seinien puhdistustehoa on tutkittu myös Itä-Suomen yliopiston Kuopion kampuksen toimesta (Tutkimusyhteistyö Itä-Suomen yliopiston kanssa alkaa 2014). Kasvien kasvualusta on täysin epäorgaaninen, joten se estää homeiden kasvun kasvualustassa. Myöskään tyypilliset mullassa munivat kasvituholaiset, kuten ripsiäiset tai villakilpikirvat eivät viihdy viherseinissä, koska ne eivät pysty munimaan mullattomassa kasvualustassa (Frequently Asked Questions About Hydroponics n.d.)

2 Valmiit ratkaisut

Valmiita asiakaspalvelu- ja vikatikettijärjestelmiä markkinoilta löytyy kymmeniä. Toimeksiantajan vaatimuksena oli hyvin toteutettu tikettiratkaisu, jolloin “help-desk”-tyyppiset ominaisuudet eivät olleet pääosassa. Vaatimuksena oli saada integroitua tiketit yrityksen omaan intranetjärjestelmään eli ulkoiseen palveluun piti päästä käsiksi jonkin rajapinnan kautta. Yksi vaihtoehto olisi ollut rakentaa koko järjestelmä alusta asti, mutta silloin olisi ollut vaikea päästä sille tasolle, jolla jo vuosia markkinoilla olleet ratkaisut ovat.

2.1 Odoo helpdesk

NaturVention Oy käyttää Odoo CRM -henkilöstönhallintajärjestelmää oman henkilöstön sekä asiakkaiden hallintaan. Odooseen on satavilla Odoo helpdesk-ja Odoo issue tracker-moduulit, jotka olivat vaihtoehtona tikettijärjestelmän pohjaksi. Tämän ratkaisun etu olisi ollut siinä, että ylläpidettäviä järjestelmiä ei olisi tullut lisää. Edellä mainituista moduuleista löytyy kuitenkin heikosti tietoa, joten niiden käyttäjiä ei to-

dennäköisesti olisi ollut kovinkaan paljon. Tämä olisi aiheuttanut sen, että järjestelmän elinkaari ei välttämättä olisi ollut riittävän pitkä. Odoo on tilattu palveluna ulkoiselta toimijalta, joten moduulien asennus olisi todennäköisesti ollut kallista.

2.2 osTicket

OsTicket on ilmainen avoimeen lähdekoodiin perustuva vikatikettiratkaisu. Se sisältää lähes kaikki maksullisten ohjelmien ominaisuudet. Siinä on RestApi rajapinta, jonka kautta tikettejä voi luoda järjestelmään. Käyttäjiä, tikettejä ja ryhmiä voi luoda loputtomasti, mikä on suuri etu, sillä niiden määrä on useimmissa muissa ratkaisuissa rajoitettu. Muiden järjestelmien kuin Os Ticketin täydestä käyttöoikeudesta täytyy usein myös maksaa. Siitä on tarjolla pilvipalveluna tarjottava versio sekä Ilmainen itsepalveluversio, jonka käyttöön kehittäjä ei tarjoa tukea. Sen käyttöönotto vaatii ohjelmistotekniikan asiantuntijaa, koska palvelinohjelmistot sekä tietokanta täytyy asentaa itse. Päivityksiä ohjelmistoon tulee harvoin, ja niiden sisältö on osin yhteisön vastuulla. OsTicket mainostaa olevansa suosituin avoimen lähdekoodin asiakaspalvelu ja vikatikettijärjestelmä, joten tukea on varmasti laajan käyttäjäkunnan vuoksi luvassa vuosiksi. OsTicket valittiin järjestelmän pohjaksi, koska sen edullisuus vei sen muiden ratkaisujen edelle.

3 Valmiit palvelut

Remo on NaturVention Oy:n oma intranetpalvelu. Sen ensisijainen tarkoitus on toimia huoltoryhmän apuvälineenä sekä viherseinien etähallintapalveluna. Remosta voi seurata seinien tilaa ja automatisointia sekä seinien lähettämää dataa veden määrästä, ilmakosteudesta sekä lämpötiloista. Sen kautta voi muuttaa seinän asetuksia ja esimerkiksi kasteluiden, valojen ja tuuletusten asetuksia voidaan säätää tarvittaessa.

Huoltohenkilökunta tekee Remoon huoltokäynneillä kirjanpitoa tekemistään töistä täyttämällä huoltolokeja. Huoltohenkilökunta voi myös muuttaa tarvittaessa seinän automaation asetuksia. Huoltolokeista voidaan seurata vaihdettujen kasvien menekkiä, kasvisairauksien esiintymistä sekä lannoitteiden ja kasvilääkkeiden kulutusta. Remossa on myös muita edistyneitä ominaisuuksia.

Remossa voi suunnitella huoltohenkilön päivittäisen huoltoreitin aikataulun, josta järjestelmä koostaa ajo-ohjeet ja päivän työlistan. Tiketit tulevat osaksi Remoa ja niillä seurataan työtehtävien valmistumista sekä pidetään huoli siitä, että työtehtävät kuitataan tehdyksi.

4 osTicket

4.1 Yleistä

OsTicket on ohjelmoitu PHP-ohjelmointikielellä ja se käyttää MySQL-tietokantaa datan varastointiin. Järjestelmävaatimuksena on Apache- tai IIS-palvelin, PHP 5.3 tai uudempi sekä MySQL 5.0 tai MariaDB. OsTicketin lähdekoodi löytyy githubista ja siitä voi halutessaan tehdä oman haaran.

Yhdysvaltalaislähtöinen OsTicketin kehitystiimi tarjoaa koulutuksia järjestelmän käyttöön ja käyttöönottoon. Saatavilla on myös etähallittu OsTicketin ylläpitämä tiketti-alusta. OsTicketin lähdekoodin käyttö on ilmaista. Ilmaislevityksessä oleva versio sisältää kaikki samat toiminnollisuudet kuin valmiiksi ylläpidetty versio.

4.2 Käyttöönotto

Asennus tapahtuu lataamalla haluttu OsTicketin versio palvelimelle. Tässä työssä osTicket asennettiin Debian palvelimelle, joten tämä ohje koskee asennusta Linux-ympäristössä. Kansion sisältämä uploads-kansio siirretään palvelimelle polussa /var/www/ tai /srv/www/ olevaan kansioon, jonka käyttäjä on nimennyt mieleisekseen. Sopiva nimi on esimerkiksi osticket tai support. Asennusohjelma tarvitsee kirjoitus- sekä lukuoikeudet, jotka annetaan /osticket/include-kansiossa olevalle ost-config.php-tiedostolle komennolla sudo chmod 755 ost-config.php. Nyt asennusohjelman voi käynnistää yhdistämällä selaimella osoitteeseen, jonka ensimmäinen osa on palvelimen IP-osoite, tässä tapauksessa 127.0.0.1/osticket/setup. Ohjelma kertoo, ovatko asennukseen tarvittavat luku-ja kirjoitusoikeudet kunnossa. Asennusohjelmaan täytetään muun muassa ylläpitäjän käyttäjänimi ja salasana sekä osTicketin tietokantaa koskevat tiedot. Asennus viimeistellään poistamalla asennusohjelman luku- ja kirjoitusoikeudet ja poistamalla /osticket/setup-kansio.

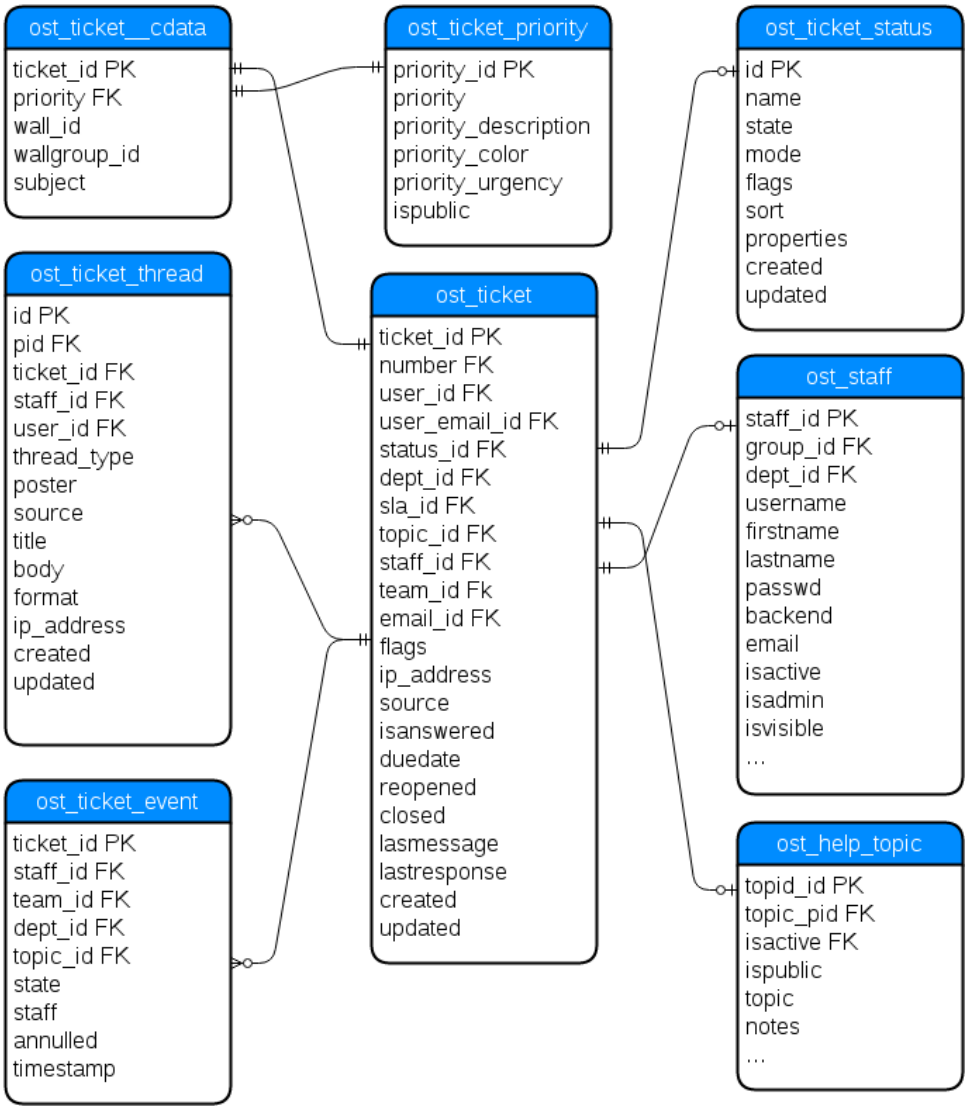
4.3 NGINX

Haasteellisin puoli asennuksessa oli saada osTicket toimimaan NGINX-palvelinohjelmiston kanssa, koska osTicketissä ei ole NGINX-tukea. Se oli kuitenkin vaatimuksena toimeksiantajalta, koska heidän kaikki palvelimensa pyörivät NGINX:n päällä.

Konfigurointi tapahtui siten, että NGINX-asetustiedostoon lisättiin säännöllisiä lauseita tunnistava lohko kaikille halutuille poluille, joita haluttiin palvella. Mikäli etsittyä polkua ei löytynyt, palautti palvelin oletuslohkoon asetetun vastauksen. OsTickettiä varten asetuksiin piti lisätä REST-rajapinnalle sekä Ajaxille omat asetuslohkonsa, joihin löytyi apua NGINX:n kotisivuilta.

4.4 Tietokanta

OsTicketin tietokanta rakentuu kymmenistä tauluista, joista osa on tarkoitettu osTicketin asetusten, käyttäjien ja muiden ominaisuuksien tallentamiseen. Tietokanta ei ole erityisen hyvin suunniteltu, koska taulujen suhteita ei voi päätellä taulujen rakenteesta. Myöskään taulujen perusavaimia ja viiteavaimia ei ole nimetty loogisesti. Tietokannan viite-eheys pidetään yllä PHP-koodissa, sillä tietokantarakenteessa ei ole määritelty vyörytyksiä eikä viite-eheyttä. Tämä johtuu todennäköisesti osin siitä, että käyttäjän oletetaan käyttävän tikettejä osTicketin kautta. Kuviossa 1 on esitetty tietokantarakenne, johon on lisätty oletettu viite-eheys.



Kuvio 1. Tiketti osTicketin tietokannassa

4.5 Organisaatio

OsTicketissä voi määritellä osastoja, joiden alla voi olla käyttäjäryhmiä ja käyttäjiä eli agentteja. Jokaiselle osastolle voi määritellä vastuuhenkilön, joka saa kaikista tiketeistä muistutuksen. Tällä henkilöllä voi olla myös admin eli ylläpitäjäoikeudet ryhmän käyttäjiin.

NaturVention Oy:n henkilökunnalle luotiin agentit osTickettiin. Jokaiselle agentille määriteltiin oletusosasto ja kohdemaa. OsTickettiin luotiin NaturVention Oy:n organisaatiorakennetta vastaavat käyttäjäryhmät. Kaikille kohdemaille tehtiin osastot myynti-, huolto- sekä tutkimus-ja kehitysosastoille.

Järjestelmään voi luoda ryhmiä, joilla hallitaan käyttöoikeuksia. Esimerkiksi agentti voi kuulua ryhmään "admin", jolla on luku ja kirjoitusoikeus kaikkiin osastoihin. Tällöin agentti näkee kaikkien maiden kaikki tiketit. Ryhmiä käytettiin Remo-integroinnissa hyödyksi ja sillä varmistettiin esimerkiksi se, että Suomessa nähdään vain Suomeen osoitetut tiketit ja etteivät Jyväskylän huollon tiketit näy Helsingin huollolle.

4.6 Rajapinnat

OsTicketissä on REST-rajapinta tikettien luomista varten. Lähes kaikki tiketit luodaan tämän rajapinnan kautta, ja sen käyttö vaatii osTicketin asetuksissa IP-osoitteesta luotavan tiivisteen tunnistautumista varten. Jokainen IP-osoite, josta tikettejä luodaan, tarvitsee oman avaimensa. Tunniste lisätään HTTP-pyynnön otsikkoparametreihin.

Data lähetetään HTTP-protokollan POST-metodilla JSON-muodossa. Näyte lähetettävästä datasta:

```
{

  "alert": true,
  "autorespond": true,
  "source": "API",
  "name": "Test user",
  "email": "api@osticket.com",
  "phone": "00000000000",
  "subject": "Testing API",
  "ip": "127.0.0.1",
  "wall_id": "666",
  "wallgroup_id": "66",
  "message": "MESSAGE HERE",
  "attachments": []
}
```

OsTicketin REST-rajapinta on hyvin puutteellinen, koska siinä on alustavasti tuki vain tikettien luomista varten. API ei tue tikettien lukemista, muokkaamista, kommentointia tai poistamista. Nämä toiminnallisuudet kirjoitettiin Remoon integroitavaan tikettirajapintaan. Edellä mainituista ominaisuuksista löytyi ”pull request”-pyyntöjä yhteisön toimesta eli yhteisön tekemiä valmiita toteutuksia näiden ominaisuuksien käyttämiseen (RESTful API for retrieving ticket information 2016). Valmista parannettua API:a kokeiltiin testiympäristössä mutta sitä ei käytetty, koska se ei sisältänyt kaikkia tarvittuja toiminnallisuksia ja se olisi estänyt osTicketin päivittämisen tulevaisuudessa.

Tikettejä voi luoda myös sähköpostin kautta. Kaikki asiakkaan havaitsemat viat ohjautuvat NaturVention Oy:n huoltosähköpostista tiketeiksi.

5 Työkalut ja teknologiat

5.1 Git

Git on Linus Torvaldsin tekemä versionhallintajärjestelmä, jonka avulla koodin hallinta on helpompaa. Git tukee paikallisten tai repositoriossa olevien haarojen käyttämistä, jolloin kesken ohjelmoinnin voidaan luoda uusi haara, jossa voidaan testata tai julkaista uusi ominaisuus. Suurempia ominaisuuksia varten tehdään yleensä oma haara, jotta pääkehityshaaraan voidaan palata päivitysten tai virheiden korjausten tekemistä varten. Valmiit työt työnnetään etäpalvelimelle Gitin kautta ja siihen liitetään viesti tehdystä työstä. Git tallentaa muokatut rivit, joten muutosten tarkastelu on helppoa. Git mahdollistaa palautuksen edelliseen ohjelmistoversioon ongelmatilanteissa (About n.d.)

5.2 PhpStorm

PhpStorm on tekstieditori, josta löytyvät kaikki ammattilaisten vaatimat ominaisuudet. Editorissa on koodin automaattinen täydennys ja se osaa ehdottaa määriteltyjä muuttujia ja PHP:n funktioita. PhpStormissa on ohjesivu kaikille PHP:n omille funktioille, josta näkee funktion vaatimat parametrit ja palautusarvot. Koodin voi laittaa

synkronoitumaan etäpalvelimelle aina tallennettaessa, jolloin koodi voi sijaita etäpalvelimella ja sitä voidaan muokata mistä vain.

5.3 Laravel

Laravel on PHP:n ympärille rakennettu ohjelmistokehys, jonka on kehittänyt Taylor Otwell. Otwel kehitti laravelin kyllästyttyään Codeigniter -frameworkin puuttelisuuteen. Laravelin ensimmäinen beta-versio julkaistiin 9.6.2011(History of Laravel PHP framework, Eloquence emerging 27.7.2011.) Vuonna 2015 Laravel oli sitepoint-nimisen verkkosivuston tekemän kyselyn mukaan ylivoimaisesti suosituin PHP-ohjelmistokehys (The Most Popular Framework of 2015 2015). Laravel tuo käyttäjälle MVC-mallin mukaisen arkkitehtuurimallin ja helpottaa datan käsittelyä tarjoamalla valmiin Eloquent ORM:n sekä QueryBuilder-luokan, joiden avulla datan käsittely sujuu vaivatta. Näkymissä voi käyttää Laravelin tarjoamaa Bladetemplating engine -mallintamista. Se mahdollistaa datan esittämisen turvallisesti. Blade-näkymiin voi useimmista muista PHP-ohjelmistokehyksistä poiketen myös kirjoittaa puhdasta PHP-koodia.

5.4 phpMyAdmin

PhpMyAdmin on ilmainen PHP:llä kirjoitettu selainpohjainen MySQL-tietokannan hallintaohjelmisto. Se tarjoaa graafisen web-käyttöliittymän datan käsittelyyn ja hallintaan sekä edistyneiden hakujen tekemiseen. PhpMyAdmin mahdollistaa myös perinteisten SQL-lausekkeiden suorittamisen web-käyttöliittymän kautta. Se tarjoaa myös laajan paletin muita työkaluja, kuten datan varmuuskopioinnin ja palautusmahdollisuuden useissa formaateissa (Import and export n.d). Tässä työssä phpMyAdminia käytettiin tietokantojen hallintaan, koska NaturVention Oy on valinnut sen tietokantojen hallintatyökaluksi ja sen ominaisuudet koettiin riittäviksi.

5.5 PHP

PHP on laajalti käytetty avoimen lähdekoodin palvelinpuolen ohjelmointikieli, joka on kehityskaarensa aikana kehittynyt olio-ohjelmointikieleksi. Sitä käytetään dynaamisten web-sivujen luontiin, tietokantojen ja sessioiden seurantaan sekä kokonaisten

web-palveluiden rakentamiseen. PHP tukee suurta osaa tietokantapalvelimista. Se tukee esimerkiksi MySQL-, PostgreSQL-, Oracle-, Sybase-, Informix-, ja Microsoft SQL Server-tietokantapalvelimia. PHP täyttää nykyään kaikki olio-ohjelmoinnin vaatimukset, ja olio-ohjelmointi PHP:llä on todella tehokasta ohjelmistokehyksien avulla. PHP:tä käytetään websivuilta lähetettävän datan käsittelyyn. Data voi olla lomakedataa tai tiedostoja, joita voidaan tallentaa tai data voidaan palauttaa käyttäjälle käsittelyn jälkeen (PHP – Introduction n.d.)

5.6 MySQL & MariaDB

MySQL on Ruotsista lähtöisin oleva ja tänä päivänä Oraclen omistuksessa oleva suosittu avoimeen lähdekoodiin pohjautuva relaatiotietokanta ja sen hallintajärjestelmä. MySQL on nimetty perustajan Michael Wildeniuksen tyttären My:n mukaan (PHP MySQL Database N.d). MariaDB on yhteisön ylläpitämä haara MySQL tietokantapalvelimesta. Se sai alkunsa, kun kehittäjät huolestuivat siitä, miten Oraclen omistus vaikuttaa MySQL:n kehitykseen. Tunnettuja MariaDB-tietokannan käyttäjiä ovat mm. Wikipedia, Facebook ja Google (About MariaDB n.d).

6 Työn toteutus

6.1 Määrittely

Asiakkaan vaatimuksena oli yhteensopivuus nykyisten palvelinohjelmistojen kanssa, joita olivat Debian Linux palvelin, NGINX http-palvelin, MariaDB-tietokantapalvelin, PHP 5.3 sekä Laravel 5.2.

Järjestelmään tikettejä tekeviä käyttäjiä ovat automaatio, asiakas sekä NaturVention Oy:n henkilökunta. Automaation täytyy pystyä tekemään tikettejä sekä API:n että ulkopuolisen palvelun kautta. Henkilökunta käyttää tikettejä lähes poikkeuksetta Remon kautta, joten integraation tikettijärjestelmän kanssa täytyy olla niin toimiva, ettei käyttäjän tarvitse avata osTickettiä tikettien käyttämiseen. Tikettijärjestelmän pitää kuitenkin olla käytettävissä erillisenä palveluna niille, joilla ei ole käyttöoikeutta Remoon.

Järjestelmän tietoturvallisuus täytyy ottaa huomioon kehitysvaiheessa. Haitallisen koodin suorittaminen tikettiä esittäessä ja tallennettaessa täytyy estää. Kirjautuminen osTickettiin olisi toivottavaa tapahtua Google tunnuksilla Oauth-tunnistautumisen kautta. Käyttäjän täytyy olla kirjautunut Remoon luodakseen tikettejä API:n kautta.

6.2 Käyttötapaukset

Käyttötaukset kuvaavat käyttäjän ja järjestelmän suhdetta. Käyttötapauksia kuvataan usein käyttötapauskaavioilla, joissa käyttäjät yhdistetään yleensä viivoilla ellipseillä piirrettyihin ominaisuuksiin ja toimintoihin. Käyttötapauksia voidaan kuvata myös käyttäjätarinoilla, joissa kerrotaan keskustelun tavoin käyttäjän ja järjestelmän suhteesta. Käyttäjätarina on yleensä lyhyt ja yksinkertainen kuvaus ominaisuudesta, joka on kerrottu sitä halunneen käyttäjän toimesta. Se seuraa yleensä seuraavaa kaavaa: Käyttäjänä voin tehdä jotain, jotta tapahtuu jotain (User Stories n.d.)

OsTickettiä koskevat käyttäjätarinat:

- Ulkoinen järjestelmä pystyy lukemaan tikettejä osTicketistä
- Ulkoinen järjestelmä voi luoda tiketin osTickettiin
- Ulkoinen järjestelmä voi kommentoida tikettiä
- Ulkoinen järjestelmä voi sulkea tiketin
- Asiakkaana voin luoda tiketin lähettämällä sähköpostia asiakastukeen

Remoa koskevat käyttäjätarinat:

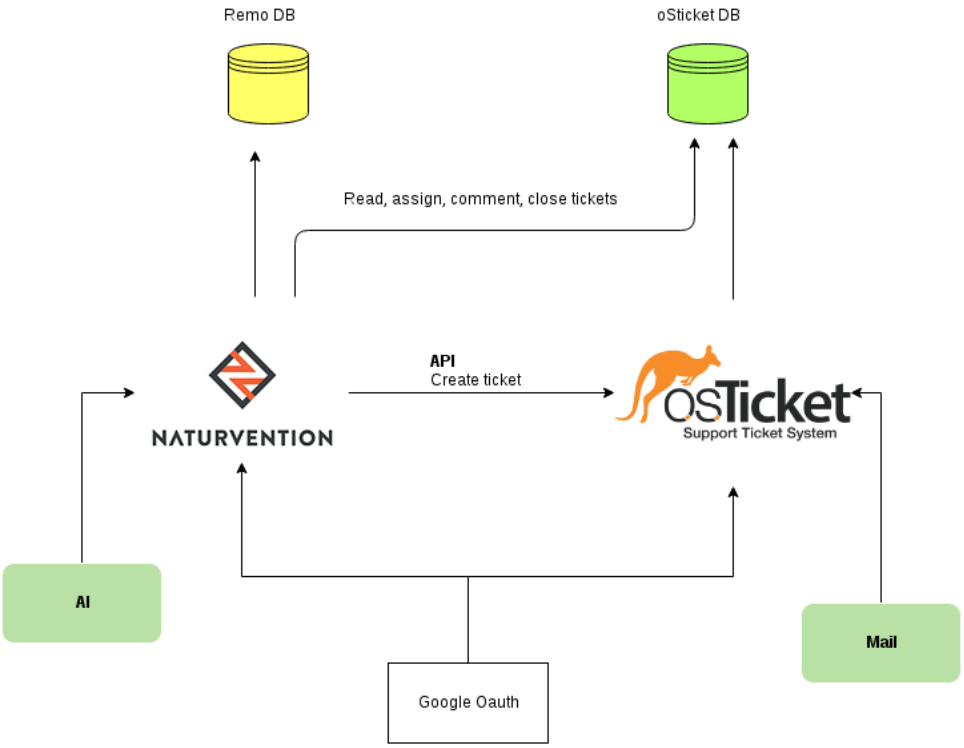
- Käyttäjänä voin lukea minulle osoitetut tiketit
- Käyttäjänä voin lukea osastolleni osoitetut tiketit
- Käyttäjänä voin lukea seinille tai seinäryhmille tehtyjä tikettejä
- Käyttäjänä voin luoda tiketin toiselle käyttäjälle
- Käyttäjänä voin luoda tiketin toiselle osastolle
- Käyttäjänä voin luoda tiketin seinälle tai seinäryhmälle
- Käyttäjänä voin kommentoida tikettiä
- Käyttäjänä voin varata tai vapauttaa tiketin varauksen
- Käyttäjänä voin sulkea tiketin
- Automaatio voi luoda tiketin seinälle tai seinäryhmälle
- Automaatio voi luoda tiketin käyttäjälle

6.3 Arkkitehtuurikuvaus

Järjestelmän arkkitehtuurissa on seitsemän hallitsevaa komponenttia (ks. kuvio 2). Sisäänkirjautuminen Remoon sekä osTickettiin tapahtuu Google-tunnuksilla. AI eli

Remoon koodattu tekoäly tarkkailee viherseiniä ja tekee tikettejä virhetilanteissa. Käyttäjä voi tehdä tikettejä joko Remon tai sähköpostin kautta. Asiakkaan ilmoittamat ongelmat ohjautuvat huoltosähköpostin kautta tiketeiksi.

Kuten kuvio 2 osoittaa, järjestelmässä on kaksi MariaDB tietokantaa: toinen on Remon datan varastointia varten (Remo DB) ja toinen osTickettiä varten (oSticket DB). Remon tietokannasta hyödynnetään käyttäjien, viherseinien ja seinäryhmien dataa, kun taas OsTicketin tietokantaa käytetään kaiken muun tiketteihin liittyvän datan varastointiin. Remon ja osTicketin välille on rakennettu silta, joka korvaa osTicketin API:n puutteellisuuden. Sen avulla voidaan lukea, kommentoida, sulkea ja varata tikettejä.



Kuvio 2. Arkkitehtuurikuvaus olemassa olevista sekä uusista järjestelmistä

6.4 Remoon tarvittavat ominaisuudet

Remoon tarvitaan ominaisuuksia, joilla voidaan hoitaa kommunikointi osTicketin kanssa. Yksi tarvituista ominaisuuksista on tikettien luominen, joka on jo tuettuna osTicketissä API:n kautta. Lisäksi tarvitaan muita toimintoja, joita ovat muun muassa tikettien lukeminen, varaaminen, vapauttaminen, kommentointi sekä sulkeminen.

Joistain tiketeistä täytyy pystyä lähettämään sähköpostimuistutus käyttäjälle. Automaation täytyy estää samanlaisten tikettien tekeminen, jotta tietokantaan ei tallenneta identtisiä tikettejä automaation toimesta.

Tikettien selaamiseen tarvitaan käyttöliittymä, josta voi selata kaikkia avoimia ja suljettuja tikettejä sekä käyttäjälle varattuja tikettejä. Näkymässä näkyvät vain tiketit, joiden lukemiseen käyttäjälle on annettu lukuoikeus. Tikettiä klikkaamalla täytyy saada esille näkymä, jossa on tarkempi kuvaus tiketistä sekä sen varaamiseen, muokkaamiseen tai sulkemiseen tarvittavat painikkeet.

Koska tiketti voi olla osoitettu viherseinälle tai seinäryhmälle sekä viherseinä- että seinäryhmäryhmäsivuilta täytyy löytyä näkymä, josta voi selata kyseiselle seinälle tai seinäryhmälle osoitettuja avoimia sekä suljettuja tikettejä.

Remossa on huoltohenkilön työjärjestyksen suunnitteluun käytettävä työkalu, josta voidaan tehdä työlista klikkailemalla kartalla näkyviä seiniä. Tähän työkaluun tarvitaan ominaisuus, joka hakee seinälle osoitetut tiketit työlistalle, jotta samalla huoltokäynnillä osataan huomioida normaalista poikkeavat työtehtävät ja jotta työmäärä seinää kohden osataan arvioida tarkemmin.

6.5 Tikettirajapinta

Tikettirajapintaan on tehty kaksitoista polkua (ks. liite 1). Jokaiselle polulle on määritelty HTTP-metodi, joita ovat POST, PUT, DELETE ja GET. HTTP-protokolla sisältää myös muita metodeja, mutta toteutuksessa on käytetty edellä mainittuja metodeja. Laravelissa polku määritellään Route -luokan metodeilla, jotka on nimetty HTTP-protokollan metodien mukaan. Metodille annetaan parametrina polku, parametrit sekä viittaus kontrolleriin sekä sen metodiin, jossa datan käsittely on toteutettu. Parametrit on määritelty polussa aaltosulkeiden sisällä. HTTP-pyyntö voi sisältää myös muita parametreja, jotka voivat olla query-parametreina eli niin sanottuina GET-parametreina URL-osoitteen lopussa tai lisättynä HTTP-pyynnön otsikkoparametreihin.

OsTicketin puuttuvat ominaisuudet eli tiketin muokkaaminen, varaaminen, vapauttaminen, sulkeminen sekä kommentointi on toteutettu TicketController-nimisessä kontrollerissa. Tikettien varaamista ja vapauttamista varten on luotu seuraavat polut:

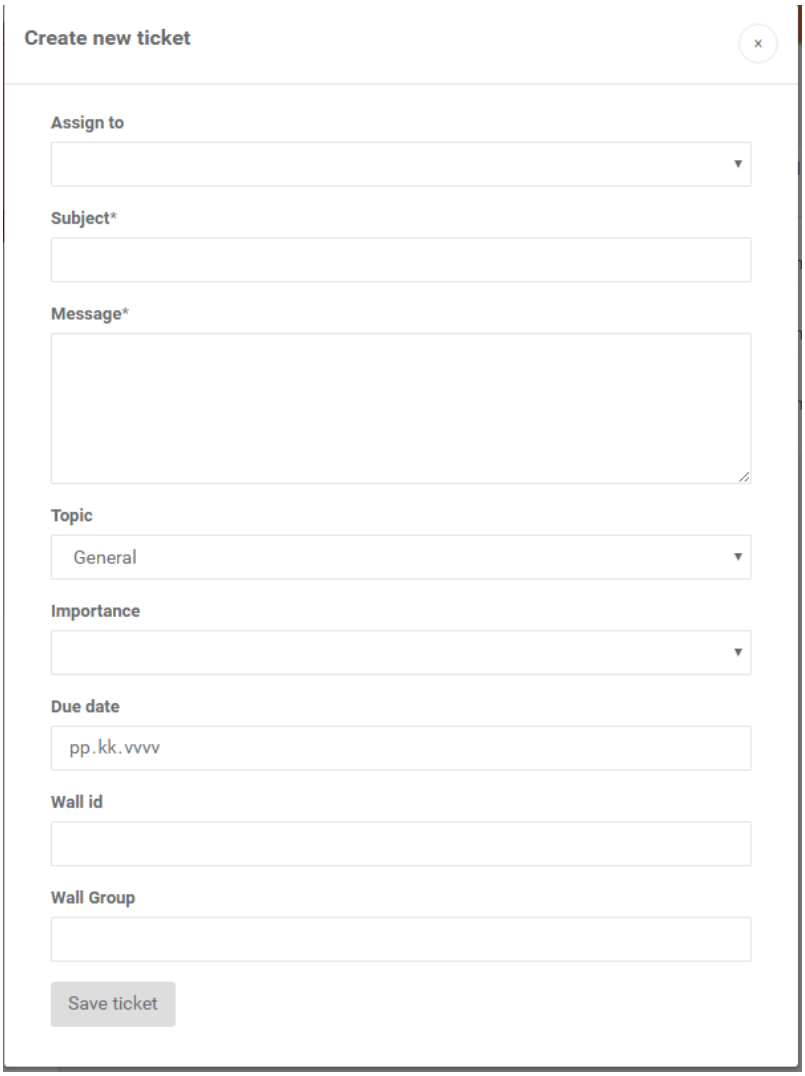
```
Route::get('/tickets/assign/{number}', 'TicketsController@assign');

Route::get('/tickets/unassign/{number}', 'TicketsController@unassign');
```

Esimerkiksi “Assign”-polku ohjaa TicketsControllerin assign-metodiin, jossa kutsutaan Ticket-nimisen luokan metodia assignTicketiksi. Ticket -luokassa luodaan osTicketin tietokannan ost_ticket taulusta olio. Ost_ticket taulun kentät ovat tämän olion ominaisuuksia. Luokassa on metodeita, joilla hoidetaan tiedon kirjoittaminen ja lukeminen osTicketin-tietokannan kanssa (ks. liite 4). OsTicketin tietokannan taulujen yhteydet olivat niin monimutkaisia, että tässä työssä ei käytetty Laravelin Eloquenttiin rakennettua toiminnollisuutta taulujen viite-eheyden rakentamiseen, vaan sen sijaan käytettiin Laravelin query builderiin rakennettuja SQL-kielisiä liitosmetodeita taulujen liittämiseen. Query-builderilla määritellään, mistä tietokannasta ja taulusta dataa haetaan. Query buideria käytettäessä paluuarvona on Laravelin ”Illuminate\Support\Collection”-luokan ilmentymä. Olion parametreihin päästään käsiksi nuolinotaatiolla object->property, tai kuten viitattaisiin taulukon tietueeseen object[”property”]. Ticket-luokkaan on kirjoitettu valmiit metodit sellaisille pyynnöille, jotka voivat toistua useassa paikassa. Tällä saadaan se hyöty, että samaa koodia ei tarvitse kirjoittaa useaan kertaan ja lisäksi data on helposti saatavissa.

Tiketin kommentointi on kuvattu liitteessä 3. Kontrollerissa validoidaan syötteet, haetaan tiketin data Ticket-luokasta sekä lisätään kommentti osTicketin ost_ticket_thread-tauluun. Käyttäjälle, jolle tiketti on varattu, lähetetään sähköpostimuistutus, jonka sisältönä on kommentti sekä kehotus toimenpiteisiin ryhtymisestä. Lopuksi sessio-muuttujiin lisätään ilmoitus käyttäjälle annettavasta palautteesta. Rajapinnan muut metodit ovat toimintatavaltaan hyvin samanlaisia. Peruskaava on seuraava: alussa validoidaan syötteet ja haetaan haluttu data. Dataa lisätään, päivitetään tai poistetaan ja lopuksi palautetaan joko dataa tai näkymä. Tikettinäkymä palautuu esimerkiksi tiketin päivittämisen jälkeen (ks. liite 2). Näkymiä palautetaan sen takia, että rajapintaa voi käyttää vain Remosta. Jos rajapinta tulisi käyttöön laajemmin, tulisi palauttaa vain dataa, jonka esittäminen hoidettaisiin frontend-puolella esimerkiksi JavaScriptin avulla.

6.6 Tikettien luominen

Tiketin luomiseen on tehty kahdeksankenttäinen lomake (ks. kuvio 3). Aihe sekä kommentti ovat pakollisia kenttiä ja loput vapaaehtoisia. Lomakkeen esitäytetyt tiedot tulevat alas vetovalikkoihin osTicketin tietokannasta. Poikkeuksena on käyttäjävalikko, jossa on osTickettiin tietokantaan käyttäjille luodut käyttäjät.



Kuvio 3. Tiketin luomiseen käytettävä lomake

Topic- eli keskustelukenttään valitaan tehtävää tai ongelmaa parhaiten kuvaava aihe. Aiheet on luotu osTickettiin (ks. kuvio 4) ja jokaiselle aiheelle on valittu vastuuhenkilö, tai osasto, jolle tiketti on osoitettu. Mikäli keskustelun lisäksi on valittu esimerkiksi käyttäjä tai korkea prioriteetti, ne ylikirjoittavat keskustelun oletusasetukset.

Showing 12 help topics — Sorting

	Help Topic	Status	Type	Priority	Department
☐	Automation	Active	**Private**	Normal	Automation
☐	Feedback	Active	Public	Low	Maintenance FI
☐	General	Active	Public	Normal	Maintenance FI
☐	Information technology	Active	Public	Normal	R&D FI
☐	Maintenance FI	Active	Public	Normal	Maintenance FI
☐	Maintenance SWE	Active	Public	Normal	Maintenance SWE
☐	Move/return request FI	Active	Public	Normal	Sales FI
☐	Plants & wall settings	Active	Public	Normal	Plants
☐	Plants production	Active	Public	Normal	Production FI
☐	Production	Active	Public	Normal	Production FI
☐	Report a Problem	Active	Public	Normal	Maintenance FI
☐	Ticketing	Active	Public	Normal	R&D FI

Select: All None Toggle

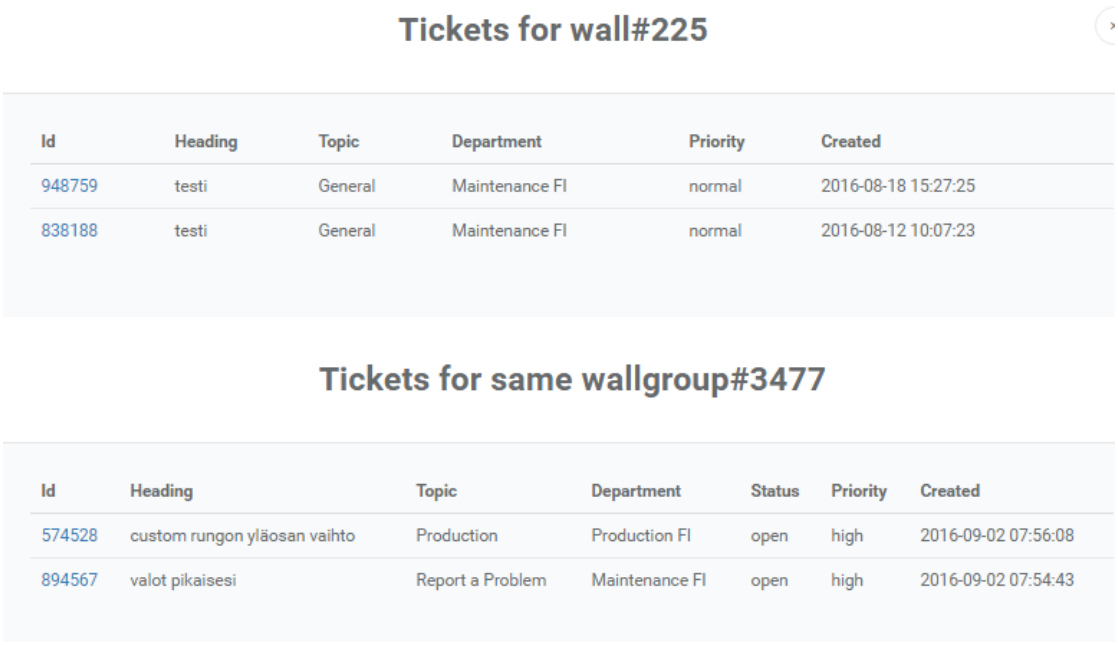
Kuvio 4. OsTicketin topicit eli aiheet

Tiketin prioriteetti voidaan valita importance- eli tärkeysasetuskentästä. Valittavissa on matala prioriteetti, normaali prioriteetti, korkea prioriteetti ja erittäin korkea prioriteetti. Jos kenttä jätetään tyhjäksi, käytetään aiheelle valittua oletusprioriteettia.

Due date -kenttään voidaan valita päivämäärä, jolloin tiketti vanhenee. Jos kenttä jätetään täyttämättä, käytetään aiheelle määriteltyä oletuspalvelutasosopimusta, joka on aiheesta riippuen 48 tai 168 tuntia. Palvelutasosopimus voidaan määritellä osTicketissä keskustelu- sekä osastokohtaisesti.

Wall id -kenttään voidaan täyttää seinän id, jolloin tiketti näkyy seinäsivulla widgetissä (ks. kuvio 5). Tiketti tulee myös näkyviin seinän sivulla olevalle tikettilistalle, jossa on kaikki seinän ja ryhmän tiketit (ks. kuvio 6). Wall group -kenttä puolestaan lisää tikettiin linkin seinäryhmän sivulle, jolloin tiketti tulee näkyviin kaikille ryhmään kuuluvien seinien sivuille.

Kuvio 5. Tiketti widgetti



Kuvio 6. Tikettilistaus seinäsivulla

Kun halutut kentät on täytetty, voidaan tallentaa tiketti. Käyttöliittymään ilmoittaa onnistuneesta tallentamisesta viestillä. Viesti tallennetaan palvelimella istuntomuuttujaan, joka poistetaan, kun viesti on näytetty.

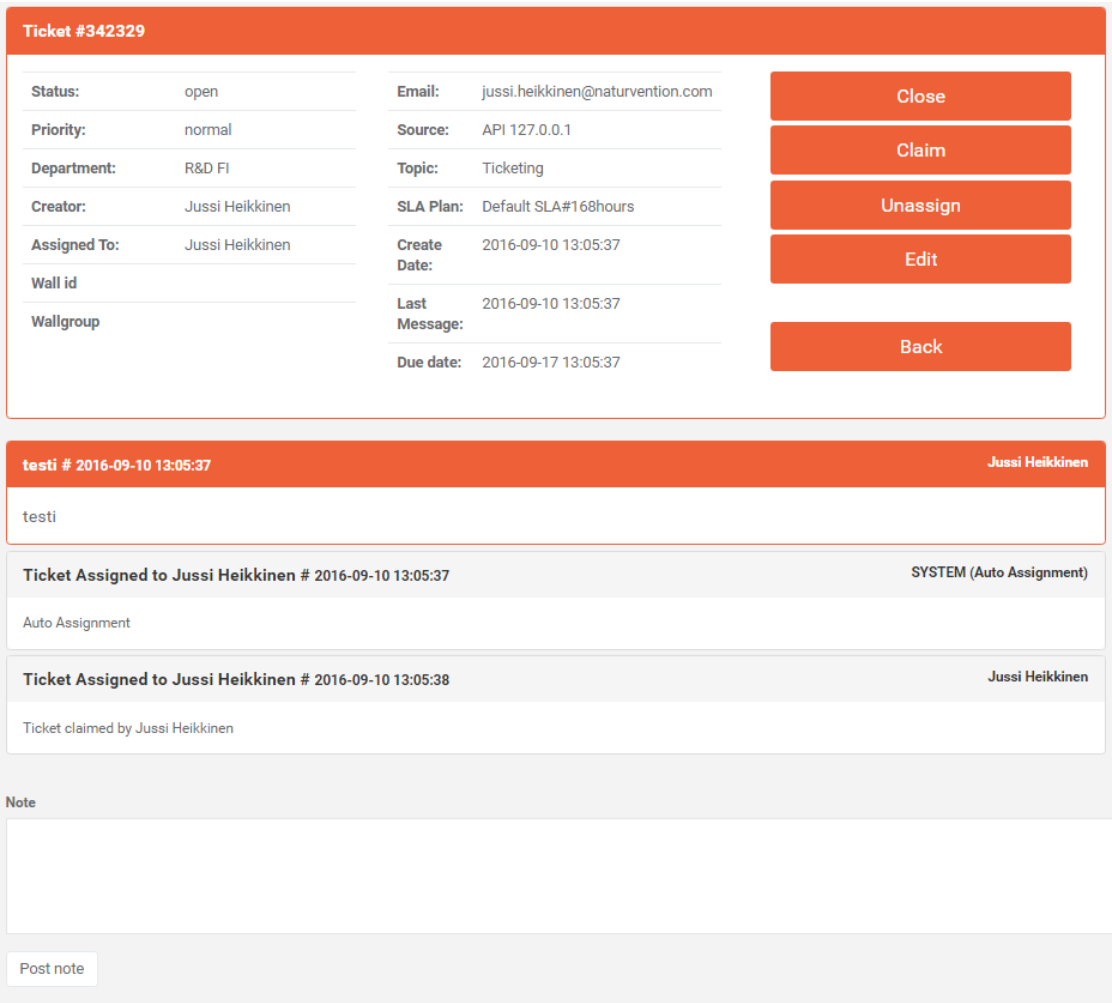
Tiketti luodaan palvelinpuolella osTicketin API:n kautta. Tiketin datasta luodaan seuraavan lainen hyötykuorma:

$payload = array("source"=>"API", "name"=>Session::get('myuser')->name, "email"=>Session::get('myuser')->email, "phone"=>Session::get('myuser')->phone, "priority"=>$request->input('importance'), "subject"=>$request->input('subject'), "topicId"=>$request->input('help_topic'), "ip"=>"127.0.0.1", "wall_id"=>$request->input('wall_id'), "instance_id"=>$iid, "wallgroup"=>$request->input('wallgroup'), "message"=>$request->input('message'), "attachments"=>[]);

Data muutetaan JSON-taulukoksi json_encode()-funktiolla ja se lisätään http- pyynnön POST- parametreihin. Pyynnön tekemiseen käytetään PHP:n curl-luokkaa. Curl olio alustetaan curl_init()-metodilla. Oliolle asetetaan halutut HTTP-parametrit curl_setopt()-metodilla. Curl suoritetaan lopulta curl_exec()-metodilla, joka tekee http-pyynnön annettuun URL-osoitteeseen.

6.7 Tikettien varaaminen, sulkeminen ja kommentointi

Tikettien varaamista, sulkemista, muokkaamista sekä kommentointia varten on tehty oma näkymä (kuvio 7). Näkymä koostuu laatikosta, jossa on esitelty tikettiin liittyvä informaatio. Laatikossa on myös tiketin muokkaamiseen käytettävät painikkeet. Laatikon alle on pinottu pienempiä laatikoita, joista ylin punaisella taustalla on tiketin viesti. Alemmat laatikot ovat tikettiin liittyviä viestejä, joita voivat olla käyttäjien tekemät kommentit ja tikettien varaamisesta ja vapauttamisesta luotavat viestit. Myös esimerkiksi osTicketin luoma ilmoitus tiketin vanhentuessa näkyy kommenttina.



Kuvio 7. Yksittäisen tiketin näkymä Remossa.

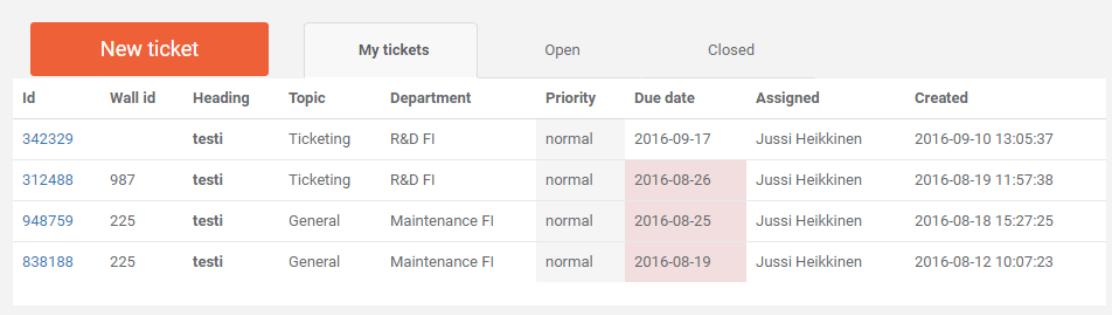
Tiketin sulkeminen tapahtuu klikkaamalla close-painiketta. Painike lataa Ajax-tekniikkaa hyödyntämällä Bootsrapin modal-ikkunaan näkymän, jossa on tekstikenttä, jota käyttäjä on pakotettu täyttämään. Tällä estetään se, että tiketti suljetaan vahingossa. Käyttäjän toivotaan kuittaavan tekstikenttään, mitä ongelmalle tehtiin. Sulkemisesta

luodaan tikettiin kommentti, jossa on sulkemisviesti. Suljetun tiketin painikkeiden käyttö estetään ja se siirretään suljettujen tikettien listalle.

Tikettiä voi muokata painamalla edit-painiketta, joka aukaisee samanlaisen lomakkeen, jolla tiketti luotiin. Myös tämän lomakkeen näkymä ladataan näkymään Ajaxin avulla. Lomakkeen kentät on esitäytetty tiedetyillä tiedoilla. Käyttäjä voi muuttaa haluttuja kenttiä ja tallentaa muutokset painamalla Save ticket -painiketta.

6.8 Käyttöliittymät

Tikettien päänäkymässä näkyvät kaikki käyttäjän varaamat tai käyttäjälle varatut tiketit (ks. kuvio 8). Näkymä koostuu kolmesta taulukosta, joista my tickets näyttää käyttäjän varaamat tai käyttäjälle osoitetut tiketit. Open-välilehdellä näkyvät kaikkien osastojen avoimet tiketit, joihin käyttäjälle on annettu lukuoikeus. Closed-välilehdellä näkyvät kaikki suljetut tiketit, joihin käyttäjällä on lukuoikeus.



Kuvio 8. Tikettien päänäkymä

Taulukon kaikki sarakkeet voidaan lajitella nousevaan tai laskevaan järjestykseen datatyypin mukaan. Lajittelu on toteutettu jQueryn tablesorter 2.0 -lisäosaa hyödyntämällä. Listat on aina oletuksena lajiteltu siten, että uusin tiketti on ylimpänä. Due date -sarakkeen tietueen tausta vaihtuu harmaasta punaiseksi, kun tiketin vanhenemispäivä tai palvelutasosopimus on ylittynyt. Prioriteettisarakkeen tietueen väri on vihreä normaalilla ja matalalla prioriteetilla. Tausta on keltainen, kun prioriteetti on korkea, ja punainen, kun prioriteetti on erittäin korkea.

6.9 Testaus

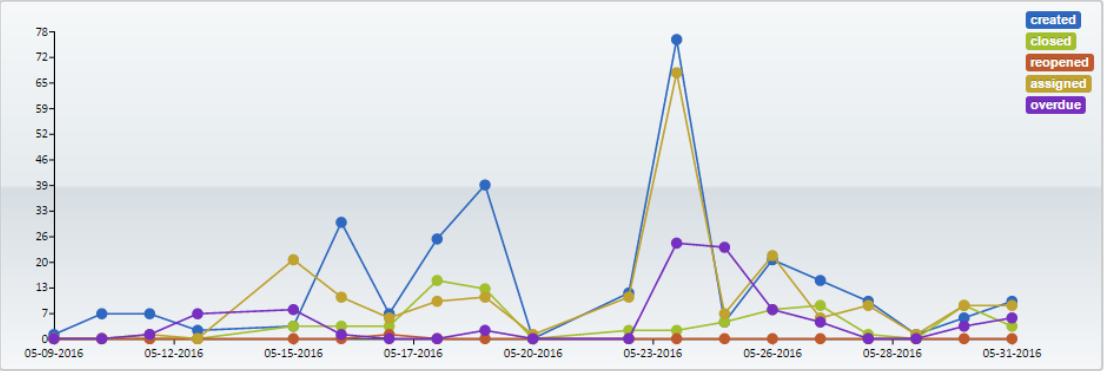
Järjestelmää on testattu kehittämisvaiheessa White-box -testausmenetelmällä, eli kehittäjänä minulla oli tieto, mitä koodissa tapahtuu. Testasin tikettien luomista, kommentointia ja muokkaamista antamalla lomakkeisiin halutun laista dataa ja tarkastamalla, tapahtuiko oletettu tapahtuma. Testasin tikettijärjestelmää myös asentamalla Google Chrome -selaimeen Bug magnet -lisäosan, jolla voi generoida yleisiä haavoittuvuuksia paljastavaa lomakedataa. Testasin Bug magnetin kautta mitä tapahtuu, kun lomake täytetään normaalista poikkeavalla datalla. Testasin sillä myös yleisimpiä haavoittuvuuksia, joita ovat muun muassa XSS eli cross-site scripting sekä SQL-injektiot, joissa lomakedata muotoillaan siten, että se saadaan suorittamaan palvelinpuolella haitallista koodia. SQL-injektiot ovat yleinen tietoturvauhka, koska niillä tunkeutuja voi tulostaa, tai poistaa dataa tietokannasta.

6.10 Käyttöönotto

Kun tikettijärjestelmä alkoi olla siinä vaiheessa, että sitä pystyi käyttämään testausympäristössä, tyhjennettiin myös lopullinen tikettijärjestelmän tietokanta testidatasta. OsTickettiin luotiin organisaatiota vastaavat ryhmät ja osastot, jotka olivat muokkautuneet lopulliseen muotoonsa.

Remon käyttäjille luotiin osTicket agentit eli käyttäjäprofiilit. Jokainen agentti sijoitettiin osastoon. Agenteille lisättiin ryhmät, joilla hallitaan agenttien lukuoikeuksia. Sähköpostiin tuleville tiketeille luotiin suodattimia, joilla viestin otsikosta tai itse viestistä pystyi säännöllisiä lausekkeita käyttäen ohjaamaan viestejä tietyille agenteille. Tikettien sähköpostimuistutukset ohjattiin siten, että osaston johtajaksi määritelty henkilö saa aina sähköpostimuistutuksen uudesta tiketistä.

Remossa oli työtehtäviä varten taulukko, josta siirrettiin tekemättömät työtehtävät tiketeiksi. Näitä tehtäviä oli 78 kappaletta ja ne näkyvät osTicketin tilastoinnissa piikkinä 23.05.2016 (ks. kuvio 9). Näillä tiketeillä saatiin järjestelmään dataa, jolla käyttäjät pääsivät tutustumaan heille osoitettuihin tiketteihin.



Kuvio 9. Remossa olevat tiketit 1.5.2016 -31.5.2016. Oikeassa yläkulmassa on selitykset viivojen väreille.

Tikettijärjestelmästä tehtiin englanninkielinen koulutusmateriaali, jossa on kerrottu tikettijärjestelmän tärkeimmät ominaisuudet sekä toiminnollisuudet. Materiaalin pohjalta tikettijärjestelmä koulutettiin Suomen huollolle sekä myöhemmin Ruotsin huollolle.

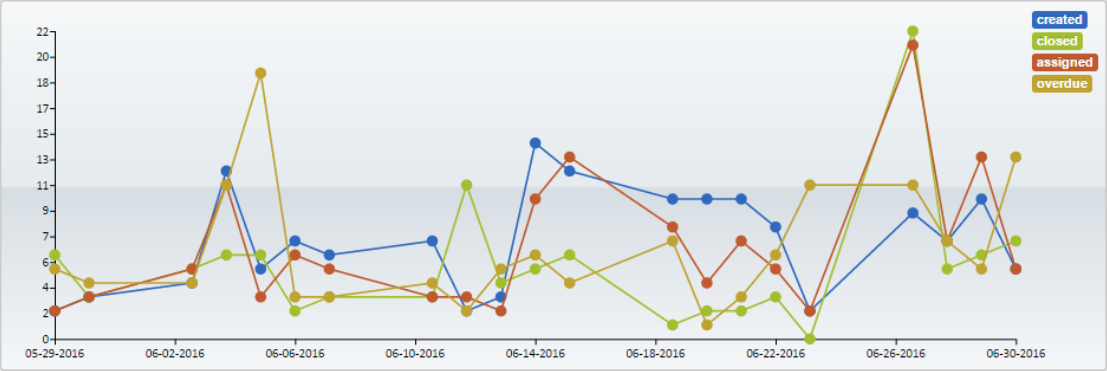
6.11 Käyttäjien palaute

Loppukäyttäjiltä palautetta järjestelmästä ei kerätty, mutta olen päässyt seuraamaan tikettijärjestelmän käyttämistä neljän kuukauden ajan ja saamani palaute on ollut positiivista. Jonkin verran parannettavaakin on tullut ilmi. Ainakin osa käyttäjistä jäi kaipailemaan henkilökohtaisia ilmoituksia tiketeistä. OsTicketissä on hyvä valikoima muistutuksia, mutta tätä asetusta ei ollut valmiina, joten se täytyy ohjelmoida rajapintaan tiketin luomisen yhteyteen.

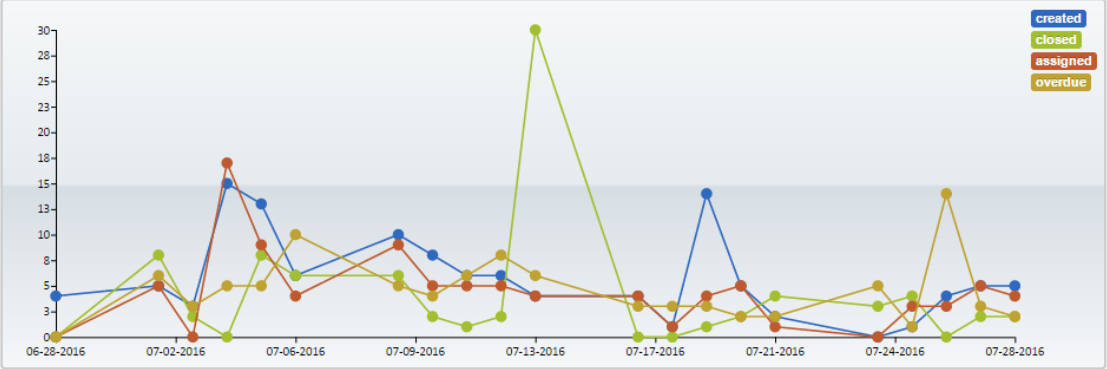
7 Pohdinta ja jatkokehitys

Tavoitteena oli saada järjestelmällinen ja luotettava järjestelmä kasvavan työmäärän hallintaan. NaturVention Oy:n henkilöstö on keskimäärin kaksinkertaistunut vuosittain, ja toimitettujen seinien määrä lähentelee tuhatta kappaletta. On selvää, että näin suuren työtaakan hallintaan tarvitaan toimiva järjestelmä, jolla voi kirjata työtehtäviä tiketeiksi, sekä ohjata nämä työtehtävät oikeille henkilöille. Järjestelmä seuraa myös sitä, että työt kuitataan tehdyksi.

Tikettijärjestelmä suoriutuu työtehtävien hallinnoinnista hyvin. Suljettuja tikettejä on neljä kuukautta käyttöönoton jälkeen 328 kappaletta ja avoimia 222 kappaletta. Kun tikettien määrä suhteutetaan tikettijärjestelmää käyttävien noin kymmenen henkilön määrään, nähdään, että tikettijärjestelmä on otettu käyttöön tehokkaasti. Käyttöä pystyy seuraamaan osTicketin tekemistä graafeista (ks. kuviot 9, 10 ja 11). Tiketeillä hallitut työtehtävät ovat yleensä jollain tavalla arkisista työtehtävistä poikkeavia vikatilanteita tai ongelmia, joten ne eivät kuitenkaan kuvaa huollon kokonaistyöpanosta.



Kuvio 10. Tiketit 1.6.2016 -30.6.2016. Oikeassa yläkulmassa on selitykset viivojen väreille.



Kuvio 11. Tiketit 1.7.2016 -31.7.2016. Oikeassa yläkulmassa on selitykset viivojen väreille.

Tavoitteena oli myös se, että automaatio osaa tehdä tikettejä. Automaatio tekee tikettejä sekä apin että sähköpostin kautta. Tähän mennessä tikettijärjestelmä on toiminut automaation kanssa hyvin, ja automaation tekemiä tikettejä on 99 kappaletta. Käyttöönottovaiheessa piti lisätä automaatiolle filttereitä, jotta samoista ongelmista ei tulisi useita tikettejä.

Yhtenä tavoitteena oli se, että tikettejä voi kohdistaa kohderyhmille, joita olivat osastot, käyttäjät, seinät ja seinäryhmät. Kaikki näihin asetetut tavoitteet saatiin toteutettua Remoon. Edellä mainituista kohderyhmistä eniten tikettejä on tehty käyttäjille ja seinille.

Tikettijärjestelmän täytyy toimia niin sulavasti osana Remoa, että käyttäjän ei tarvitse käyttää sitä osTicketin kautta, vaan kaikki toiminnot ovat käytettävissä Remon kautta. Tässä tavoitteessa onnistuttiin hyvin, sillä kaikki tiketit on tehty Remon tai sähköpostin kautta. Kaikki toimeksiantajan vaatimat ominaisuudet saatiin lisättyä Remon tikettirajapintaan.

Tikettijärjestelmää täytyy myös jatko kehittää. Jatkokehitys koostuu enimmäkseen pienistä muutoksista, joilla järjestelmä saadaan mukautumaan yrityksen tilanteeseen sopivaksi. Näitä muutoksia ovat esimerkiksi käyttäjien, osastojen sekä muistutusten päivittäminen. Suurempia ylläpitotehtäviä voivat olla osTicketin- ja Laravelin päivittämisestä mahdollisesti aiheutuvat ongelmat. NaturVentionin nopea kasvu niin Suomessa- että globaalisti aiheuttaa tikettijärjestelmälle omat haasteensa. Sen täytyy tarvittaessa skaalautua suurelle käyttäjämäärälle, sekä selviytyä kansainvälistymisestä aiheutuvista haasteista.

Lähteet

About. N.d. Git:n verkkosivut. Viitattu 28.10.2016. https://git-scm.com/about/free-and-open-source

About MariaDB. N.d. MariaDB:n verkkosivut. Viitattu 1.9.2016. https://mariadb.org/about/

Frequently Asked Questions About Hydroponics. N.d. Beylik farms:n verkkosivut. Viitattu 17.10.2016.http://www.beylikfarms.com/hydroponicsFAQs.htm

History of Laravel PHP framework, Eloquence emerging. 27.7.2013. Blogi. Viitattu 28.10.2016. https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/

Import and export. N.d. PhpMyAdminin verkkosivut. Viitattu 15.10.2016. https://docs.phpmyadmin.net/en/latest/import_export.html?highlight=export

PHP – Introduction. N.d. Tutorials point:n verkkosivut. Viitattu 17.10.2016. https://www.tutorialspoint.com/php/php_introduction.htm

PHP MySQL Database. N.d. W3 school:n verkkosivut. Viitattu 15.10.2016. http://www.w3schools.com/php/php_mysql_intro.asp

Rekisteritiedot, NaturVention Oy. 2016. Asiakastiedon verkkosivut. Viitattu 21.9.2016. https://www.asiakastieto.fi/yritykset/fi/naturvention-oy/24326611/rekisteritiedot

RESTful API for retrieving ticket information. 2016. Git pull request. Viitattu 17.10.2016. https://github.com/osTicket/osTicket/pull/2947

The Most Popular Framework of 2015. 2015. Sitepoint:n verkkosivut. Viitattu 30.8.2016. https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

Tutkimusyhteistyö Itä-Suomen yliopiston kanssa alkaa. 2014. Blogi. Viitattu 21.9.2016 https://www.naturvention.com/fi/blogi/tutkimusyhteistyo-ita-suomen-yliopiston-kanssa-alkaa/

User Stories. N.d. Mountain Goat Software:n verkkosivut. Viitattu 19.10.2016. https://www.mountaingoatsoftware.com/agile/user-stories

VOC-päästöt. N.d. Hengitysliiton verkkosivut. Viitattu 18.8.2016. http://www.hengitysliitto.fi/fi/sisailma/hiukkasmaiset-ja-kaasumaiset-epapuhtaudet/voc-paastot

Liitteet

Liite 1. Tikettirajapinnan reitit

```
Route::get('/tickets', 'TicketsController@index');
Route::post('/tickets', 'TicketsController@store');
Route::get('/tickets/{id}', 'TicketsController@showByUser');
Route::put('/tickets/{id}', 'TicketsController@update');
Route::delete('/tickets/{id}', 'TicketsController@destroy');
Route::get('/tickets/assign/{number}', 'TicketsController@assign');
Route::get('/tickets/{id}/show', 'TicketsController@showTicket');
Route::get('/tickets/unassign/{number}', 'TicketsController@unassign');
Route::post('/tickets/close/{number}', 'TicketsController@close');
Route::post('/tickets/{number}/comment', 'TicketsController@comment');
Route::get('/tickets/ajax/help/{param}', 'TicketsController@showGroupList');
Route::get('/tickets/ajax/tickets/{wid}', 'TicketsController@getTicketsForManager');
```

Liite 2. Tiketin päivittäminen

```
/**
* Update the specified resource in storage.
* PUT /tickets/{id}
*
* @param  int  $id
* @return Response
*/
public function update($id, Request $request)
{
 // return $request;
  $this->validate($request, [
      'wall_id' => 'numeric',
      'wallgroup' => 'numeric',
  ]);

  Ticket::updateTicket($id, $request, Session::get('myuser')->email);
  //Give feedback
  session()->flash('flash_message', 'Ticket '.$id.' has been updated');

  return redirect()->action('TicketsController@showTicket', [$id]);
}
```

Liite 3. Tiketin kommentointi

```
/**
* POST /tickets/comment post a comment to a ticket
*
* @return mixed
*/
public function comment($number, Request $request){
  $this->validate($request, [
      'title' => 'required',
      'note' => 'required||string',
  ]);

  $ticket = Ticket::where('number','=', $number)->join('ost_ticket__cdata', 'ost_ticket.ticket_id', '=',
'ost_ticket__cdata.ticket_id')->first();
  $ticket->lastmessage = Carbon::now('Europe/Helsinki');
  $ticket->save();
  $staff = DB::connection('osticket')->table('ost_staff')->where('email', '=', Session::get('myuser')-
>email)->first();

  if ($ticket->status_id == 3){
      Ticket::openTicket($ticket, $staff);
  }

  DB::connection('osticket')->table('ost_ticket_thread')->insertGetId(['pid'=>0, 'ticket_id'=>$ticket-
>ticket_id, 'staff_id'=>0, 'user_id'=>$ticket->user_id, 'thread_type'=>'N',
      'poster'=>$staff->firstname.' '.$staff->lastname, 'title'=>$request->input('title'), 'body'=>$request-
>input('note'),
      'format'=>'html', 'ip_address'=>$ticket->ip_address, 'created'=>Carbon::now('Europe/Helsinki'),
'updated'=>'0000-00-00 00:00:00']);

  $subject = "Your ticket $ticket->number have been commented";
  $message = "Your ticket <a href='(ip)/tickets/".$ticket->number."/show'>#".$ticket->number."</a>
have been commented. Check it in case your ticket need more information to be
solved.<br><br>Message: ".$request->input('note');
  $to = DB::connection('osticket')->table('ost_user_email')->where('user_id', $ticket->user_id)->first();
  if(preg_match("/[a-z.0-9]+[@?]naturvention\W[a-z]+/i", $to->address) || preg_match("/[a-z.0-
9]+[@?]fresh-effect\W[a-z]+/i", $to->address)){
      RemoFunctionHelper::send_email($message, $to->address, $subject);
  }
  //Give feedback
  session()->flash('flash_message', 'Your comment have been created');

  return redirect()->action('TicketsController@showTicket', [$number]);
}
```

Liite 4. Tiketin luominen

```
/**Create ticket using osTicket API
*
* @param $payload
* @param $apikey
* @return mixed
*/
public static function createTicket($payload, $apikey)
{
  $ch = curl_init();

  $header = array("X-API-Key: " . $apikey, "Content-Type: multipart/form-data");
  curl_setopt($ch, CURLOPT_URL, 'osticketin URL');
  curl_setopt($ch, CURLOPT_POST, true);
  curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
  curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($payload));
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  //curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_ANY);
  //curl_setopt($ch, CURLOPT_USERPWD, "user:pass");
  $curlresult = curl_exec($ch);

  if ($curlresult === false) {
     return false;
  }

  curl_close($ch);

  return $curlresult;
}
```

Liite 5. Osa Ticket-luokasta

```
namespace App\Models;
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;
use Session;
use Naturvention\Remo\RemoFunctionHelper;

class Ticket extends Model
{
  //
  protected $connection = 'osticket';
  protected $table = 'ost_ticket';
  protected $primaryKey = 'ticket_id';
  public $timestamps = false;

  public function ostUser(){
     return $this->hasOne('App\Models\Ost_user');
  }

  public function ostStaff(){
     return $this->hasOne('App\Models\Ost_staff');
  }

  public static function getTicketsByWall($iid,$closed){
     $closed == true ? $closed = 3 : $closed = 1; //closed is 3 in database 6 does not exist

     return Ticket::where('ost_ticket.status_id', '=', $closed)//Not closed
        ->where('ost_ticket__cdata.instance_id', '=', $iid)
        ->join('ost_ticket__cdata','ost_ticket.ticket_id', '=', 'ost_ticket__cdata.ticket_id')
        ->join('ost_user', 'ost_ticket.user_id', '=', 'ost_user.id')
        ->join('ost_department','ost_ticket.dept_id','=','ost_department.dept_id')
        ->join('ost_ticket_status','ost_ticket.status_id','=','ost_ticket_status.id')
        ->join('ost_help_topic','ost_ticket.topic_id','=','ost_help_topic.topic_id')
        ->select('ost_ticket.*',
           'ost_ticket__cdata.priority','ost_ticket__cdata.instance_id','ost_ticket__cdata.wall_id',
'ost_ticket__cdata.subject',
           'ost_user.name',
           'ost_help_topic.topic',
           'ost_department.dept_name',
           'ost_ticket_status.name','ost_ticket_status.state')
        ->orderBy('created', 'desc')
        ->get();
  }

  public static function getTicketsByUser($email){
     $staff = DB::connection('osticket')->table('ost_staff')->where('email', '=', $email)->first();

     return Ticket::where('ost_ticket.status_id', '<>', 3)
        ->where('ost_ticket.staff_id',"=", $staff->staff_id)
        ->join('ost_ticket__cdata','ost_ticket.ticket_id', '=', 'ost_ticket__cdata.ticket_id')
        ->join('ost_user', 'ost_ticket.user_id', '=', 'ost_user.id')
        ->join('ost_department','ost_ticket.dept_id','=','ost_department.dept_id')
        ->join('ost_ticket_status','ost_ticket.status_id','=','ost_ticket_status.id')
        ->join('ost_help_topic','ost_ticket.topic_id','=','ost_help_topic.topic_id')
        ->select('ost_ticket.*',
           'ost_ticket__cdata.priority','ost_ticket__cdata.wallgroup','ost_ticket__cdata.instance_id',
'ost_ticket__cdata.wall_id', 'ost_ticket__cdata.subject',
           'ost_user.name', 'ost_user.org_id',
           'ost_help_topic.topic',
           'ost_department.dept_name',
           'ost_ticket_status.state')
        ->orderBy('created', 'desc')
        ->get();
  }
```

```
public static function getTicketsByWallgroup($wallgroup, $iid, $closed){
    $closed == true ? $closed = 3 : $closed = 1;

    return Ticket::where('ost_ticket.status_id', '=', $closed)//Not closed
        ->where('ost_ticket__cdata.instance_id', '<>', $iid)
        ->where('ost_ticket__cdata.wallgroup', '=', $wallgroup)
        ->join('ost_ticket__cdata', 'ost_ticket.ticket_id', '=', 'ost_ticket__cdata.ticket_id')
        ->join('ost_user', 'ost_ticket.user_id', '=', 'ost_user.id')
        ->join('ost_department','ost_ticket.dept_id','=','ost_department.dept_id')
        ->join('ost_ticket_status','ost_ticket.status_id','=','ost_ticket_status.id')
        ->join('ost_help_topic','ost_ticket.topic_id','=','ost_help_topic.topic_id')
        ->select('ost_ticket.*',
            'ost_ticket__cdata.priority','ost_ticket__cdata.wallgroup','ost_ticket__cdata.subject',
            'ost_user.name',
            'ost_help_topic.topic',
            'ost_department.dept_name',
            'ost_ticket_status.name','ost_ticket_status.state')
        ->orderBy('created', 'desc')
        ->get();
}

public static function getAllTicketsWithData($name, $closed, $access_table){

    $closed ? $closed = 3 : $closed = 1; //closed is 3 in database 6 does not exist

    $table = array();
    foreach ($access_table as $access){
        array_push($table, $access->dept_id);
    }

    return Ticket::where('ost_ticket.status_id', '=', $closed)
        ->whereIn('ost_ticket.dept_id', $table)
        ->join('ost_ticket__cdata', 'ost_ticket.ticket_id', '=', 'ost_ticket__cdata.ticket_id')
        ->join('ost_user', 'ost_ticket.user_id', '=', 'ost_user.id')
        ->join('ost_department','ost_ticket.dept_id','=','ost_department.dept_id')
        ->join('ost_ticket_status','ost_ticket.status_id','=','ost_ticket_status.id')
        ->join('ost_help_topic','ost_ticket.topic_id','=','ost_help_topic.topic_id')
        ->select('ost_ticket.*',
            'ost_ticket__cdata.priority','ost_ticket__cdata.wallgroup','ost_ticket__cdata.instance_id',
'ost_ticket__cdata.wall_id','ost_ticket__cdata.subject',
            'ost_user.name', 'ost_user.org_id',
            'ost_help_topic.topic',
            'ost_department.dept_name',
            'ost_ticket_status.state')
        ->orderBy('created', 'desc')
        ->get();
}

public static function getTicketByIdWithData($id){

    return Ticket::where('number','=',$id)
        ->join('ost_ticket__cdata', 'ost_ticket.ticket_id', '=', 'ost_ticket__cdata.ticket_id')
        ->join('ost_help_topic','ost_ticket.topic_id','=','ost_help_topic.topic_id')
        ->join('ost_user', 'ost_ticket.user_id', '=', 'ost_user.id')
        ->join('ost_sla', 'ost_ticket.sla_id', '=', 'ost_sla.id') //tämä karsii suljetut tiketit tuloksesta koska
sla taulussa ei ole idtä 0
        ->join('ost_department','ost_ticket.dept_id','=','ost_department.dept_id')
        ->join('ost_ticket_status','ost_ticket.status_id','=','ost_ticket_status.id')
        ->join('ost_user_email', 'ost_ticket.user_id', '=', 'ost_user_email.user_id')
        ->select('ost_ticket.*',
            'ost_ticket__cdata.priority','ost_ticket__cdata.wallgroup','ost_ticket__cdata.in-
stance_id','ost_ticket__cdata.wall_id','ost_ticket__cdata.subject',
            'ost_sla.grace_period', 'ost_sla.name as slaname',
            'ost_user.name as username',
            'ost_department.dept_name',
```

```
            'ost_help_topic.topic',
            'ost_ticket_status.state','ost_ticket_status.state',
            'ost_user_email.address')
        ->first();
  }
}
```