SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

TIKETTIJÄRJESTELMÄN LUONTI TOIMINNANOH-JAUSJÄRJESTELMÄÄN

TE-KIJÄ: Teemu Lätti

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Teemu Lätti	
Työn nimi Tikettijärjestelmän luonti toiminnanohjausjärjestelmään	
Päiväys 11.12.2018	Sivumäärä/Liitteet 31
Ohjaaja(t) Lehtori Jukka Kinnunen, Lehtori Sami Lahti	
Toimeksiantaja/Yhteistyökumppani(t) NettiTieto Oy	

Tiivistelmä

Tämä opinnäytetyö toteutettiin toimeksiantona suomalaiselle IT-alan yritykselle NettiTieto Oy:lle. Opinnäytetyön aiheena on tikettijärjestelmän luonti toiminnanohjausjärjestelmään. Työn tavoitteena oli perehtyä tikettijärjestelmän vaatimuksiin ja luoda sellainen osaksi NettiTiedon omaa toiminnanohjausjärjestelmää. Tikettijärjestelmä oli ajankohtainen, sillä tikettijärjestelmää oli toivottu toiminnanohjausjärjestelmää käyttävien asiakkaiden toimesta.

Tämä opinnäytetyö on kehittämistyö toimeksiantajalle koostuen teoria-osasta ja tikettijärjestelmästä. Teoria-osuus sisältää tietoa käytetyistä tekniikoista, tikettijärjestelmän vaatimuksista ja muista huomioiduista asioista. Koodaus tapahtui PHP-ohjelmointikielellä, Laravel-ohjelmointikehystä käyttäen. Sähköpostilaatikon lukemiseen käytettiin IMAP-protokollaa ja tietokantakyselyt suoritettiin Laravelin Eloquentia hyödyntäen.

Opinnäytetyön lopputuloksena on toimiva tikettijärjestelmä osana toiminnanohjausjärjestelmää, jonka avulla pystytään luomaan tikettejä ja vastaamaan niihin. Tikettijärjestelmä tulee jatkossa osaksi toiminnanohjausjärjestelmää. Tikettijärjestelmää tullaan kehittämään jatkossa lisää tarpeiden mukaisesti.

Avainsanat
Tikettijärjestelmä, Laravel

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Teemu Lätti			
Title of Thesis Creating Ticketing System into ERP-System			
Date	11 December 2018	Pages/Appendices	31
Supervisor(s) Mr Jukka Kinnunen, Lecturer and Mr Sami Lahti, Lecturer			
Client Organisation /Partners NettiTieto Oy			

Abstract

This thesis was commissioned by a Finnish IT company NettiTieto Oy. The topic of this thesis was to create a ticketing system into the ERP system. The ticketing system was topical because it was asked for by the users of NettiTieto's enterprise resource planning system.

The first target was to become familiar with the needs of ticketing. The ticketing system needed to read e-mails from the mailbox and create tickets or comments from e-mails. The second target was to make commenting and browsing tickets possible. The third target was to send comments by e-mail to the receiver.

This thesis was a development work for the comissioner consisting of related theory and the ticketing system. The theory was collected from e-books, blogs and trusted websites. NettiTieto's coding standard was to use PHP scripting language alongside with the Laravel framework. Reading mailbox was made with the IMAP protocol and database queries were called by Laravel's own object relational mapper Eloquent. Sublime Text 3 was used as a text editor.

As a result of thesis a functional ticketing system was made that can create tickets and respond to them. After solving a ticket it can be closed. In future the ticketing system will be a part of the enterprise resource planning system and it will be developed by the coming needs.

Keywords
Ticketing system, Laravel

SISÄLTÖ

1 JOHDANTO 6
1.1 Toimeksiantaja ja opinnäytetyön esittely 6
1.2 Käsitteet 6
1.3 Lyhenteet ja määritelmät 6
2 ITIL – INFORMATION TECHNOLOGY INFRASTRUCTURE LIBRARY 8
2.1 Määritelmä & historia 8
2.2 ITIL-malli 8
3 TIKETTIJÄRJESTELMÄ 10
3.1 Tikettijärjestelmä yleisesti 10
3.2 Tikettijärjestelmän toiminnot 10
3.3 Tiketin käsittely 11
4 WEB-KEHITYS 12
4.1 Web tekniikat 12
4.2 Käytetyt tekniikat 13
4.2.1 MVC 14
4.2.2 Versiohallinta 15
4.2.3 Tietokanta 17
4.2.4 Testaus 17
5 SELVITYSTYÖ 18
5.1 Selvitystyön kohteet 18
5.1.1 IMAP 18
5.1.2 POP 18
5.2 OWASP 19
5.2.1 SQL-injektio 21
5.2.2 XSS 21
6 TYÖN TOTEUTUS 22
6.1 Suunnittelu 22
6.2 Toteutusvaiheet 22
6.3 Lopputulos 24
7 POHDINTA 27

8 JATKOKEHITYS 28

LÄHTEET 29

1 JOHDANTO

1.1 Toimeksiantaja ja opinnäytetyön esittely

Opinnäytetyön toimeksiantajana toimi NettiTieto Oy, joka on suomalainen vuonna 2003 perustettu ohjelmisto- ja viestintäratkaisuja tarjoava palveluyritys (NettiTieto Oy). Opinnäytetyön aiheena oli tikettijärjestelmän luonti osaksi NettiTiedon tarjoamaa toiminnanohjausjärjestelmää. NettiTiedolla oli tarve tikettijärjestelmälle, koska toiminnanohjausjärjestelmästä puuttui tikettijärjestelmä ja asiakkaat olivat sitä toivoneet.

Uudelle järjestelmälle löytyi heti pitkä lista erilaisia ominaisuuksia ja vaatimuksia. Kokonaisen järjestelmän luonnin katsottiin olevan aivan liian laaja opinnäytetyöksi, joten aihetta rajattiin. Opinnäytetyön tavoite oli luoda toimiva pohja järjestelmälle, joka osaa muodostaa sähköposteista tikettejä. Tiketistä täytyi tarkistaa onko sillä tikettitunniste ja jos tikettitunniste löytyi, liitetään se alkuperäiseen tikettiin kommenttina. Jos taas tiketillä ei ole tunnistetta, muodostuu siitä uusi tiketti. Sähköpostilaatikosta halutaan lukea uudet viestit automaattisesti, joten täytyi tehdä ajastettu toiminto. Sähköpostit poistetaan sähköpostilaatikosta automaattisesti sen jälkeen, kun niistä on muodostettu tiketti tai kommentti.

Tikettien muodostuessa automaattisesti sähköposteista täytyi miettiä millaisia uhkia siihen sisältyy. Tähän liittyen tutustuin OWASP:in kymmeneen yleisimpään tietoturvariskiin vuodelta 2017.

1.2 Käsitteet

Tikettijärjestelmä	Järjestelmä, jonka avulla hallinnoidaan tuki/palvelupyyntöjä.
Tiketti	Yksittäinen tukipyyntö, jota voidaan hallinnoida.

1.3 Lyhenteet ja määritelmät

Git	Versionhallintaohjelmisto
HTML	Hypertext Mark Language on yleisimmin käytetty merkintäkieli
IMAP	Internet Message Access Protocol on sähköpostiviestien hallinnointiin käytetty protokolla
IT	Information Technology, yleisnimitys tietotekniikasta
ITIL	Information Technology Infrastructure Library, kokoelma parhaita käytäntöjä IT-palveluiden johtamiseen ja hallintaan
Laravel	MVC-pohjainen web-sovelluskehys, joka on kirjoitettu PHP-kielellä
MVC	Model-View-Controller, arkkitehtuurimalli, joka jaottelee sovelluksen loogisiin osiin tehden siitä tehokkaamman

PHP	Hypertext Preprocessor, avoimen lähdekoodin skriptikieli web-kehitykseen, jota voidaan upottaa HTML-kielen sekaan
POP	Post Office Protocol on sähköpostiviestien hallinnointiin käytetty protokolla
SQL	Structured Query Language, tietokantakieli, jota käytetään tietokannan hallintaan ja tiedon manipulointiin
XML	Extensible Markup Language, merkkikieli
XSS	Cross Site Scripting, hyökkäys, jossa ohjelmakoodia sisällytetään HTML:n sekaan

2 ITIL – INFORMATION TECHNOLOGY INFRASTRUCTURE LIBRARY

2.1 Määritelmä & historia

ITIL on lyhenne sanoista Information Technology Infrastructure Library® ja sillä tarkoitetaan kokoelmaa IT-palveluiden parhaista käytännöistä ja menetelmistä (White ja Greiner, 2017.) ITIL sai alkunsa 1980-luvulla osana Iso-Britannian hallituksen hanketta. Ensimmäinen kirjasto oli käytössä vuodet 1989–1996. ITIL v2-version kehitys alkoi 1990-luvun lopussa. Samoihin aikoihin syntyi palveluhallinnan BS15000-standardi, joka kehittyi myöhemmin ISO 20000-standardin ensimmäiseksi versioksi. Vuonna 2000 julkaistiin ITIL Service Support v2 valtion kauppaministeriön toimesta. Vuosien 2000–2001 välillä julkaistiin yhdeksän kirjasettiä, joista suosituimpia olivat tuki ja palvelusetit. ITIL v3 julkaistiin toukokuussa 2007 koostuen viidestä eri kirjasta. Tämä versio vahvisti nykyisen ITIL-mallin. Samoihin aikoihin julkaistiin useita palveluhallinnon viitekehyksiä jotka mallintivat ITIL-perusteita. (Gisuser, 2018.)

Teknologian kehittyessä vauhdilla vuonna 2011 huomattiin tarve päivitetylle versiolle. ITIL:n nykyinen omistaja AXELOS ilmoitti vuoden 2017 loppupuolella uuden ohjeistuksen julkaisusta 2018 (Gisuser 2018.)

2.2 ITIL-malli

ITIL:n tavoitteita on selkeyttää asiakkaan ja työntekijän välistä keskustelua, vähentää kuluja, selkeyttää IT-tuen toimintatapoja, parantaa tuottavuutta, kehittää taitoja ja kokemusta, helpottaa liiketoiminnallista riskienhallintaa ja rakentaa vakaan ympäristön, joka on silti muutettavissa.
ITIL-elinkaari koostuu viidestä eri tilasta, jotka ovat palvelustrategia, palvelusuunnitelma, palvelun muutos, palvelun valvonta ja jatkuva palvelun kehittäminen (Greycampus.)

Palvelustrategia on keskeisin osa, koska se määrittelee mitä palveluita organisaatio tarjoaa ja mitä taitoja tulee kehittää. Tärkeimpinä palvelustrategian aihealueita ovat markkinoiden määrittely, strategisten varojen kehittäminen, käyttöönoton valmistelu ja tarjoukset.
Palvelusuunnittelun tehtävänä on suunnitella IT-strategia ja tehdä parannuksia olemassa olevaan suunnitelmaan. Palvelusuunnitteluun sisältyy seuraavia aihealueita: palveluhakemisto, palvelutasot, kapasiteetti, saatavuus, palvelun jatkuvuus, tietoturva ja toimittajien hallintaprosessit. (Irwin, 2018.)

Palvelun muutos on organisaation uuden suunnittelun rakennus, testaus ja toteutusvaihe. Seuraamalla näitä vaiheita ja korjaamalla esiintyvät ongelmat yritys varmistaa, ettei palveluissa ole häiriöitä. Palvelun muutokseen kuuluvat muutoksen suunnittelu ja tuki, muutosten hallinta, julkaisun ja käyttöönoton hallinta, palvelun validointi ja testauksen arviointi ja tiedonhallintaprosessi. (Irwin, 2018.)

Palvelun operointi valvoo uuden palvelun käyttöä oikeassa ympäristössä. Uusia ongelmia ilmaantuu yleensä kun palvelu siirtyy asiakkaan käyttöön, joten palvelua täytyy valvoa ja olla valmiina korjaamaan ongelmat, vastaamaan asiakkaan pyyntöihin, korjaamaan palvelun viat ja suorittaa rutiininomaisia operatiivisiä tehtäviä. (Irwin, 2018.)

Jatkuvalla palvelun kehittämisellä tarkoitetaan, ettei organisaation pitäisi koskaan tyytyä nykyiseen palvelun laatuun, sillä aina tulee uusia ongelmia joita kehittää. ITIL:n viimeinen vaihe ohjeistaa organisaatioita identifioimaan ja analysoimaan mahdolliset kehityskohteet neljässä edeltävässä kohdassa. Jatkuvan kehityksen kannalta organisaatioiden tulisi seurata seitsemän kohdan kehitysprosessia, johon kuuluu määritellä mitä mitataan, määritellä mitä ei mitata, kerätä dataa manuaalisesti tai mittaustyökaluilla, käsitellä dataa, analysoida dataa, esittää data sopivalle yleisölle ja toteuttaa parannustoimet. (Irwin, 2018.) Kaikki viisi edellä mainittua vaihetta on esitelty kuviossa 1.

KUVIO 1. Viisi ITIL tasoa. (Irwin, 2018).

3 TIKETTIJÄRJESTELMÄ

3.1 Tikettijärjestelmä yleisesti

Tikettijärjestelmällä tarkoitetaan järjestelmää, jonka avulla hallinnoidaan service deskiin tulevia tuki/palvelupyyntöjä. Tikettijärjestelmiä on olemassa paljon. Niitä on tarjolla sekä ilmaisina että maksullisina versioina. Tunnettuja tikettijärjestelmiä ovat muun muassa Zendesk, Otrs, Handesk ja OsTicket.

Yrityksellä, joka käsittelee asiakkaidensa tukipyyntöjä, on hyvä olla käytössään tikettijärjestelmä. Tikettijärjestelmän avulla tiketit pysyvät samassa muodossa rippumatta siitä kuka niitä on käsitellyt. Jokaisesta asiakkaan antamasta tukipyynnöstä muodostetaan tiketti, jota hallinnoidaan tarvittavilla toimenpiteillä. Tikettijärjestelmän avulla on sujuvampaa seurata tukipyyntöjen suorituksia. Suorituksista voidaan muodostaa raportteja halutulta ajanjaksolta. Raporttien pohjalta on helppo seurata millaisia tukipyyntöjä tulee eniten, jolloin saadaan palautetta mahdollisista puutteista ohjedokumentaatioissa.(Spier, Vincent, Foley, Rolsky & Chamberlain 2008.)

Tiketillä tarkoitetaan yksittäistä tukipyyntöä tai ongelmaa. Tukipyyntö voi olla vaikka unohtunut salasana tai käyttöoikeuksien määrittäminen tietylle henkilölle. Ongelma puolestaan voi olla esimerkiksi estynyt pääsy verkkoon tai ettei verkkosivut toimi totutulla tavalla. Tiketti voidaan määrittää prioriteettiasteikoilla kuinka akuutti se on. Sille voidaan antaa omistaja, eli henkilö joka hallinnoi tikettiä. Tikettiin voidaan vastata ja sille voidaan antaa status. Tiketissä täytyy olla myös vastaanottoaika, muokkausaika, tikettitunniste ja otsikko.

Tiketti voidaan muodostaa monella eri tapaa. Opinnäytetyön toimeksiantajan tapauksessa halutaan, että tiketti voidaan muodostaa sähköpostiin tulleesta tukipyynnöstä, web-sivuilla olevan palautelomakkeen kautta, web-sivuilla olevan yhteydenottopyynnön kautta, toisesta sovelluksesta sekä järjestelmän sisältä käsin.

3.2 Tikettijärjestelmän toiminnot

Tikettijärjestelmän keskeisiä toimintoja ovat yleisesti tiketin rekisteröinti, tiketin vastuuhenkilön/omistajan määritys, usean osapuolen määritteleminen tikettiin, tiketin muutosten kirjaus, muutosten informointi tiketin osapuolille, toiminnon suoritus perustuen tiketin statukseen tai prioriteettiin, raportointi statuksen perusteella ja tiketin sulkeminen (Spier, Vincent, Foley, Rolsky & Chamberlain 2008).

3.3 Tiketin käsittely

Tikettiä käsitellessä tavoitteena on suorittaa toimenpiteet, joilla ongelma korjataan. Siihen voidaan kommentoida asiakkaan sekä työntekijän toimesta. Tikettiin voidaan myös liittää toinen työntekijä, jolla voi olla aikaisempaa kokemusta samankaltaisesta ongelmasta. Muita mahdollisia liitoksia tiketille ovat toinen, jo olemassa oleva tiketti ja liitos mahdolliseen projektinhallintajärjestelmään. Tiketeille voidaan antaa tageja, eli sanoja jotka kuvastavat ongelmaa parhaiten. Nämä tagit helpottavat jatkossa valmiiden vastausten etsimisessä, kun ilmenee vastaavia ongelmia. Lopuksi tiketti suljetaan, kun se on ratkaistu. Ratkaistut tiketit on hyvä säilyttää tallessa, jotta tulevaisuudessa vastaavat onglemat voidaan ratkaista hetkessä.

4 WEB-KEHITYS

4.1 Web tekniikat

Web-sivusto koostuu yleisesti merkkikielestä, etupalvelimesta(front-end), taustapalvelimesta (back-end) ja tietokannoista. Merkkikielen tehtävänä on esittää web-sivustolla näkyvät asiat. Etupalvelin puolestaan suorittaa etupalvelimella tapahtuvan koodin. Taustapalvelimen tehtävä on suorittaa yhteydenotto tietokantaan ja datan käsittelyä vaativat toiminnot. Merkkikielellä tarkoitetaan tietokonekieltä, joka päättelee käytettävät elementit tagien avulla. Ihminen kykenee lukemaan merkkikieltä. Kaksi suosituinta merkkikieltä ovat HTML ja XML, jotka molemmat ovat luettavissa yleisillä tekstieditoreilla. (The Tech Terms Computer Dictionary.)

Front-end etupalvelimen avulla saadaan tehtyä web-sivusta interaktiivinen käyttämällä erilaisia ohjelmointikieliä tai ohjelmointikehyksiä. Etupalvelimelle koodaaminen tarkoittaa yleensä sivuston ulkoasuun liittyviä ratkaisuja. Etupalvelimen ohjelmoinnissa käytettyjä ohjelmointikieliä ovat mm. JavaScript, VBScript ja ActionScript. Ohjelmointikehyksistä ja kirjastoista suosittuja ovat mm. jQuery, AngularJS, React, Vue ja Ember. (Wodehouse.)

Back-end taustapalvelin on se osa joka toteuttaa sivuston toiminnot. Taustapalvelimella on yleensä suora yhteys tietokantaan. Taustapalvelimen ohjelmoinnissa käytettyjä ohjelmointikieliä ovat mm. PHP, C#, Java ja Python. Suosittuja taustapalvelimen ohjelmointikehyksiä ovat mm ASP.NET, Node.js, Laravel ja Ruby on Rails. (Wodehouse.)

Seuraava esimerkki esittää web-sivuston toimintaa ravintolan avulla.
Ajatellaan ravintolan pöytää etupalvelimena ja keittiötä taustapalvelimena. Asiakkaan saapuessa ravintolaan ohjataan asiakas pöytään. Vastaavasti nettisivuille saapuessa käyttäjä ohjataan etusivulle. Etupalvelimella käyttäjä voi lukea HTML-merkkikielellä kirjoitettua sivustoa, ravintolassa voidaan tarkkailla ravintolan ulkoasua ja ruokalistaa, jonka voidaan ajatella olevan HTML-merkkikielellä kirjoitettu. Etupalvelimella voi olla Javascript-koodilla kirjoitettuja toimintoja jotka eivät vaadi yhteyden ottamista taustapalvelimelle, ravintolassa esimerkiksi tarjoilija voi toimia Javascript-koodina. Verkkosivulla voi olla kirjautumislomake, joka vaatii kirjautumistunnusten lähettämisen taustapalvelimelle, joka tarkistaa tunnukset ja uudelleenohjaa käyttäjän kirjautuneen käyttäjän näkymään. Ravintolassa tilataan ruoka lähettämällä pyyntö tarjoilijalle joka palauttaa keittiöstä vastauksena tilatun ruuan. (Kononenko 2018.) Tähän esimerkkiin voidaan lisätä vielä tietokanta. Taustapalvelin on yleensä yhteydessä tietokantaan ja ravintolassa tietokannassa voidaan ajatella olevan vaikka reseptit joita kokit hyödyntävät.

4.2 Käytetyt tekniikat

Opinäytetyö toteutettiin PHP-ohjelmointikielellä, jonka apuna käytettiin Laravel-ohjelmointikehystä. Tietokantaohjelmistona käytettiin MariaDB:ta ja sitä hallinoitiin HeidiSQL:n avulla. Ulkoisten kirjastoiden latausta varten käytettiin Composeria ja Packagistiä. Versiohallinta tapahtui Git:in ja GitLabin avulla.

PHP on ilmainen avoimeen lähdekoodiin perustuva taustapalvelimella suoritettava ohjelmointikieli ja se on helppo oppia. Se kehitettiin vuonna 1994 Rasmus Lerdorfin toimesta. PHP soveltuu käytettäväksi myös web-kehitykseen ja sitä voidaan sulauttaa HTML-merkistön sekaan. (PHP.net). PHP koodi suoritetaan palvelimella ja tulokset palautetaan selaimelle puhtaana HTML-merkistönä. (w3schools).

PHP-kielen avuksi kehitetty Laravel-ohjelmointikehys perustuu MVC-arkkitehtuuriin. Se kehitettiin vuonna 2011 Taylor Otwellin toimesta, kun Otwell kyllästyi CodeInteger-ohjelmointikehyksen ongelmiin ja päätti ratkaista ne luomalla kokonaan uuden ohjelmointikehyksen. (O'Brien, 2016.)

Eloquent on Laravelin ActiveRecord-toteutus. Yhteydenpito tietokantaan olio-orientoituneesti on nykystandardi. Eloquentin avulla tietokannan luonti, poisto, päivitys ja haku tietokannasta onnistuvat kirjoittamatta riviäkään SQL-koodia. Lisäksi Eloquent pystyy hoitamaan monia metodeja kuten sivujen numeroinnin automaattisesti. (McCool, 2012.)

Artisan on Laravelin komentorivin työkalu, jonka avulla voidaan suorittaa muun muassa migraatioita, testejä ja ajastettuja tehtäviä. (McCool, 2012). Controllerien, modelien ja mm. seederien luominen onnistuu myös näppärästi Artisan komennolla. Esimerkkikomento voisi olla vaikka: *$ php artisan make:controller MailController*, joka luo uuden MailController nimisen controllerin.

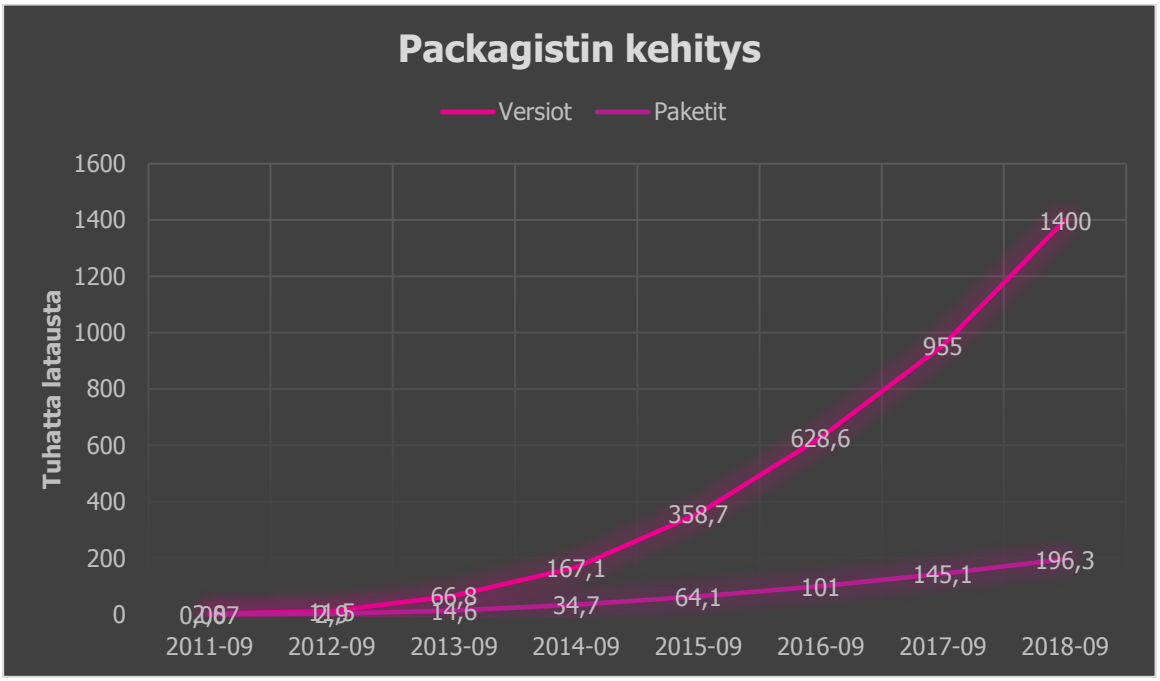
Laravelissa on valmiiksi asennettuja kirjastoja, jotka tarjoavat hyödyllisiä lisäominaisuuksia kuten esimerkiksi autentikointi-kirjasto. Muutosten teko tietokannan rakenteisiin on helppoa Laravel migraatioiden avulla, eikä tarvitse kirjoittaa riviäkään SQL-koodia vaan kaikki hoituu PHP-kielellä. Laravel huolehtii tietoturvasta käyttämällä esim. salasanojen tallennukseen moderneja salausalgoritmeja. Laravelin Unit-testaus on kehittäjien keskuudessa erittäin pidetty ominaisuus (Bhagat 2017.)

Composer on työkalu riippuvuuksien hallintaan PHP-ohjelmointikielellä. Sen avulla määritetään kirjastot, joista projekti on riippuvainen. Composer osaa myös asentaa ja päivittää kirjastot automaattisesti. Toimiakseen Composer vaatii vähintään PHP-version 5.3.2 (getcomposer.org.)

Composerin käyttämistä varten täytyy Composer olla asennettuna, jonka jälkeen projektiin tarvitsee lisätä vain composer.json-tiedosto, johon määritetään mihin kirjastoihin halutaan riippuvuudet. Composer osaa etsiä kirjastot packagist.org sivustolta sekä ladata ne ja päivittää niitä. Composer lataa paketit oletuksena projektikohtaisesti, ei globaalisti. (Saunier 2014.)

Composerin käyttö on nopeampaa kuin pakettien manuaalinen etsiminen, lataus ja asentaminen. Käyttämällä Composeria vältetään ristiriidat versioiden välillä, kun päivitetään yksittäisiä riippuvuuksia. Eri luokat ladataan automaattisesti Composerin toimesta. Composerin tavoittelee yhtä keskeistä pakettien säilytyspaikka, josta paketit haetaan ja ladataan (Saunier 2014.)

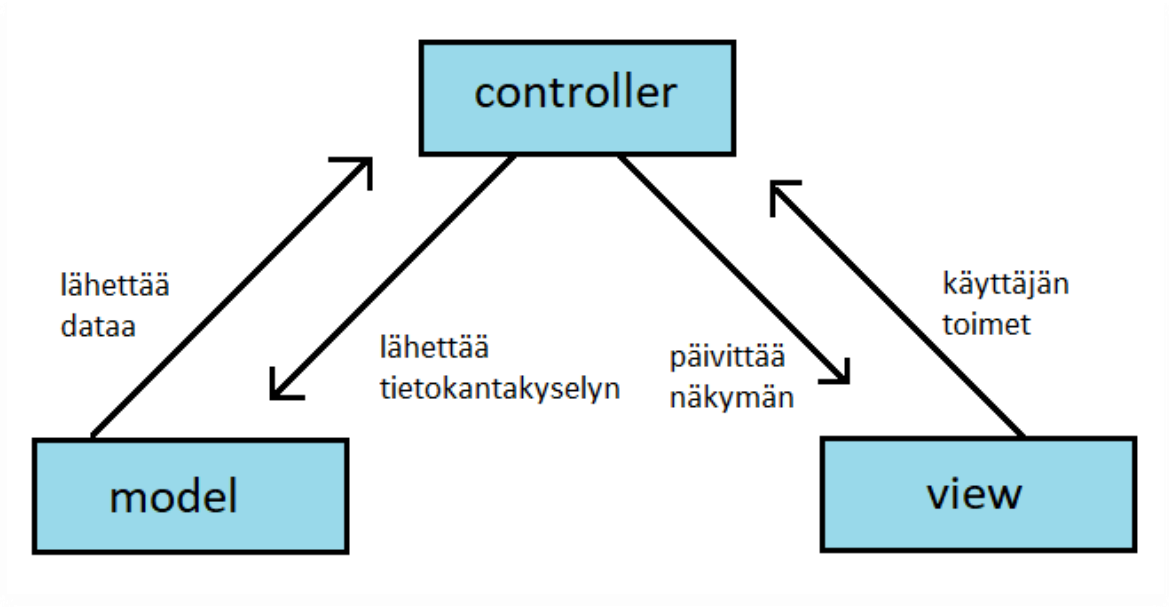
Packagist on Composerin tärkein säilytyspaikka, josta ladataan paketit projektia varten. Packagistin tavoitteena on olla keskeinen säilytyspaikka, jota kaikki käyttävät (getcomposer.org). Vuonna 2011 Packagistissä oli alle 50 pakettia ja alle 3000 versiota, nyt lokakuussa 2018 pakettaja on melkein kaksisataatuhatta ja versioita melkein 1,4 miljoonaa. Packagistiin voi myös ladata omia paketteja muiden käyttäjien käyttöä varten. (Kuvio 5).



KUVIO 5. Packagistin pakettien ja versioiden kehitys (Packagist.com).

4.2.1 MVC

Model-View-Controller arkkitehtuurilla tarkoitetaan sovelluksen jakamista kolmeen pääkomponenttiin. Model eli malli esittää dataa, jota siirrellään näkymän ja ohjaimen välillä. View eli näkymä esittää selaimella näkyvän sivun ja siihen liittyvät asiat kuten tekstilaatikot, kuvat, värit jne. Controller eli ohjain toimii näkymien ja mallien käsittelyssä esimerkiksi käsittelemällä lomakkeen näkymässä ja päivittämällä tietokannan hyödyntämällä mallia. MVC-mallissa View hoitaa etupalvelimen näkymää, Model toimii taustapalvelimella hoitaen yhteyden tietokantaan ja Controller keskustelee edellä mainittujen komponenttien välillä. (Kuvio 2).



KUVIO 2. MVC-arkkitehtuurin malli (Ighodaro 2018).

MVC-mallin etuja ovat sen jakautuminen useaan eri osaan, jolloin yksi henkilö voi keskittyä kokonaan näkymien koodaamiseen ja toinen henkilö voi keskittyä controllerien logiikan koodaukseen. MVC mahdollistaa URL-osoitteiden täydellisen hallinnan, kun jokaisen osoitteen voi määritellä itse reittien avulla.

4.2.2 Versiohallinta

Versiohallinta tarkoittaa järjestelmää, joka pitää kirjaa kaikista sovelluksen koodiin tehdyistä muutoksista. Näin ollen tarvittaessa voidaan palata takaisin tiettyyn versioon. Sen avulla pystyy seuraamaan kuka on muokannut mitäkin tiedostoa, vertailla muutoksia eri versioiden välillä sekä palauttaa tiedostoja tiettyyn, aikaisempaan tilaan. (Chacon & Straub 2014.)

Linux-käyttöjärjestelmän kehittäjä Linus Torvalds, kehitti versionhallinta-järjestelmä Git:in pitääkseen miljoonat rivit koodiaan selkeässä hallinnassa (Santacroce 2015, 1). Gitin hyödyt tulevat esiin, kun samaa projektia päivittävät useat eri henkilöt. Versionhallinnan avulla on helppo selvittää kuka on tehnyt mitäkin ja virhetilanteissa on helppo palata viimeiseen toimivaan versioon.

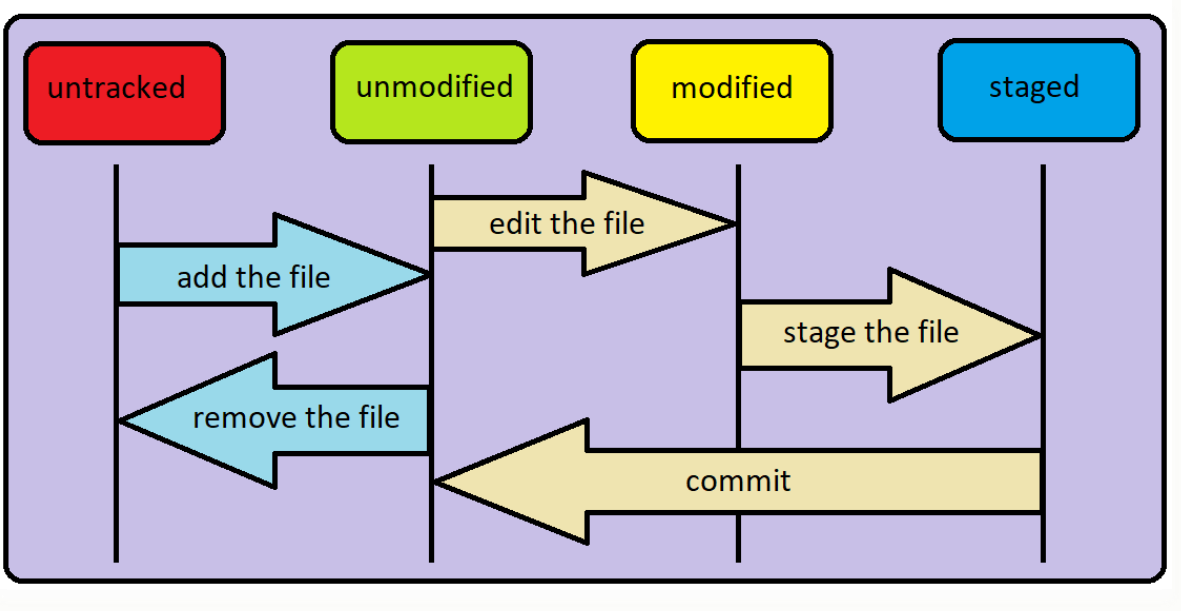
Git-projektilla on kolme erilaista osaa. Nämä ovat työskentelyhakemisto, Git-hakemisto ja Valmistelualue. Työskentelyhakemistossa tehdään muutoksia paikallisesti, jonka jälkeen muutokset siirretään valmistelualueelle odottamaan siirtoa pysyvään muutokseen. Työskentely Gitin parissa alkaa yleensä muokkaamalla tiedostoja työskentelyhakemistossa. Tämän jälkeen valmistellaan tiedostot, lisäämällä niistä tilannekuvat (snapshot) valmistelualueelle. Lopuksi tehdään pysyvä muutos, jolloin tiedostot valmistelualueelta varastoidaan pysyvästi Git-hakemistoon. (Santacroce 2015, 5-6.)

Kun Git on asennettu tietokoneelle, voidaan aloittaa sen käyttäminen. Ensin valitaan projekti, jossa halutaan käyttää Git:iä. Tämä tapahtuu joko kloonaamalla olemassa oleva Git-tietolähde (repository)

palvelimelta tai aloittamalla jäljittää olemassa olevaa projektia hakemistosta. Kloonaus tapahtuu komentorivi-komennolla *$ git clone [url]*. Kun taas halutaan aloittaa olemassa olevan projektin jäljitys, mennään komentorivillä kyseisen projektin hakemiston juureen ja ajetaan komento *$ git init*. Komennolla *$ git add [tiedosto]* aloitetaan tiedoston jäljitys. Kun jäljitetty tiedosto halutaan lisätä osaksi Git-hakemistoa, ajetaan komennot *$ git commit "Add tiedosto.txt"* ja *$ git push origin master.* Ensimmäinen komento tekee "Add tiedosto.txt"-nimisen commitin, joka sisältää tiedostot jotka on lisätty add-komennolla. Toinen komento "työntää" Git-hakemistoon kaikki commitit, jotka se löytää työskentelyhakemistosta. (Santacroce 2015, 6-7.)

Git perustuu kolmeen pääasialliseen tiedoston tilaan: valmisteltu (staged), muutettu (modified) ja pysyvästi muutettu (commited). Näistä ensimmäisenä mainittu tarkoittaa, että muutettu tiedosto on merkattu nykyisessä versiossaan menemään seuraavaan pysyvään tilannekuvaan. Muutetulla tarkoitetaan tiedostoa jota on muutettu, muttei ole vielä tehty pysyvää muutosta. Pysyvästi muutetulla tiedostolla tarkoitetaan, että data on varastoitu turvallisesti tietokantaan. (Chacon & Straub, 2014.)

Alla olevassa kaaviossa on lisäksi kuvattu tilat unmodified ja untracked. Untracked tila kertoo, että projektissa on tiedosto, jota Git ei jäljitä. Tiedosto saa tämän tilan, kun luodaan uusi tiedosto tai poistetaan tiedosto Git-repositorysta. Tiedoston tilan saa helpoiten selville ajamalla komento *$git status*. (Kuvio 3).



KUVIO 3. Git statuksen elinkaari (Chacon 2009).

4.2.3 Tietokanta

Tietokannalla tarkoitetaan paikkaa, johon tieto tallennetaan selkeäksi kokonaisuudeksi ja josta se on helposti saatavilla. ”Tietokanta on tietojen keräämiseen ja järjestämiseen käytettävä työkalu. Tietokantoihin voidaan tallentaa esimerkiksi henkilöihin, tuotteisiin ja tilauksiin liittyviä tietoja. Monet tietokannat ovat aluksi tekstinkäsittelyohjelmalla luotuja luetteloita tai laskentataulukoita.”(Microsoft Office.)

MariaDB on yksi maailman käytetyimpiä tietokantapalvelimia. Sen ovat kehittäneet MySQL:n kehittäjät ja sen taataan pysyvän avoimen lähdekoodin tietokantana. MariaDB:n etuja ovat sen nopeus, skaalattavuus, varmuus ja runsas määrä liitännäisiä ja työkaluja. (MariaDB.)
HeidiSQL on ilmainen sovellus tiedon ja rakenteiden hallintaan. Se soveltuu käytettäväksi MySQL, MariaDB, Microsoft SQL ja PostreSQL–tietokantapalvelimille. (HeidiSQL.)

4.2.4 Testaus

Koodin laadun varmistuksessa on fiksuinta asettaa tietyt standardit, joita seurata. Yrityksellä tai organisaatiolla on usein käytössä omat standardit koodauskäytäntöihin, jotta vältytään syntaksi eroilta ja jotta kuka tahansa yrityksessä voi sujuvasti lukea tai muokata koodia. Standardien avulla koodi näyttää siistiltä eikä ole sekoitus erinäköistä koodia. Tiivistettynä onnistuneet standardit seuraavat seuraavia ohjelinjauksia: hyödyntävät yhteisön ohjeistuksia, automatisoivat ohjeiden noudattamista, säilötään versionhallintaan sekä ovat helppoja lukea ja seurata.

Test-driven development (TDD) on kehitysmenetelmä joka painottaa testin kirjoittamista lisäosalle ennen sen toteuttamista. (Kuvio 4).

1	Lisää testi
2	Aja kaikki testit ja tarkista epäonnistuuko uusin testi
3	Kirjoita koodi uuteen testiin
4	Aja testit
5	Muokkaa koodia
6	Toista

KUVIO 4. Test-driven develpment prosessin kulku. (Scott 2017.)

Yksikkötestauksessa koodia testataan hyvin pienissä paloissa, käytännössä vain yhtä toimintoa kerrallaan. Yksikkötestaus koostuu määrittelemällä mitä metodia testataan, kutsumalla kyseistä metodia ja tarkastelemalla tuloksia. Hyvä yksikkötesti on nopea, hyvin kirjoitettu ja helposti ymmärrettävissä. Hyvin kirjoitetun yksikkötestin avulla voidaan korjata virhe koodista kokonaan ilman koodin debuggausta. Testien ei pitäisi olla riippuvaisia ulkoisista tekijöistä tai testien suoritusjärjestyksestä. (Kolodiy.)

5 SELVITYSTYÖ

5.1 Selvitystyön kohteet

Opinnäytetyötä varten täytyi selvittää sopiva protokolla sähköpostilaatikon lukemista varten. Tietoturvauhkien selvityksessä perehdyttiin OWASP:in määrittelemiin yleisimpiin uhkakuviin. Eräänä työn tavoitteena oli sähköpostilaatikon lukeminen tikettijärjestelmästä. Tutkittaessa sähköpostilaatikon lukua nousi esiin kaksi eri menetelmää; IMAP ja POP3. Alla olevissa kaavioissa (Kuvio 6 & Kuvio 7) on havainnoitu molempien menetelmien hyvät ja huonot puolet.

5.1.1 IMAP

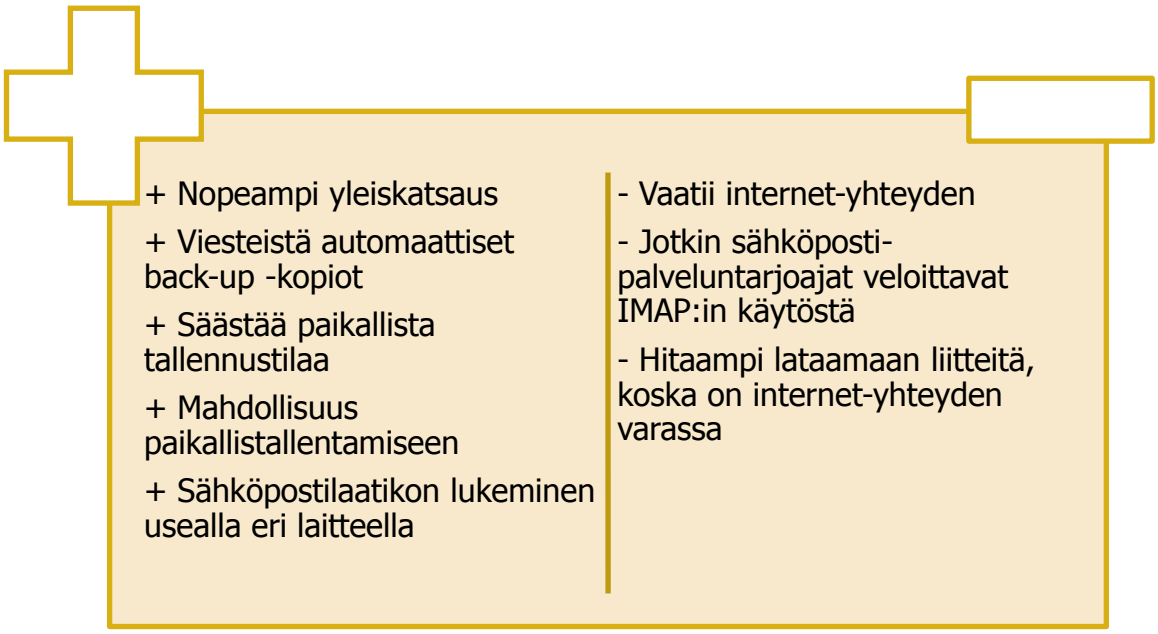
IMAP on sähköpostien hakuun käytettävä protokolla, joka on kehittyneempi vaihtoehto POP3-protokollalle. IMAP:in suurin etu POP3:en nähden on viestien tallennus postipalvelimelle sen sijaan, että kaikki ladataan paikalliselle koneelle. Tämä on tärkeää käyttäjille, jotka käyttävät sähköpostiaan useilla eri laitteilla. (Dean 2009, 502–503.)

IMAP mahdollistaa kaikkien viestien tai vain osan viesteistä vastaanoton, mikä on hyödyllistä käytettäessä hidasta nettiyhteyttä tai kun käytössä on minimaalisesti tilaa kovalevyllä. Myös viestien poistaminen nopeutuu, kun ei tarvitse ladata pitkiä viestejä tai isoja liitteitä paikalliselle koneelle ennen poistamista. IMAP-palvelimet vaativat enemmän tallennustilaa ja resursseja kuin POP3. Jos IMAP-palvelin kaatuu, eivät käyttäjät pääse käsiksi viesteihin. (Dean 2009, 502–503.)

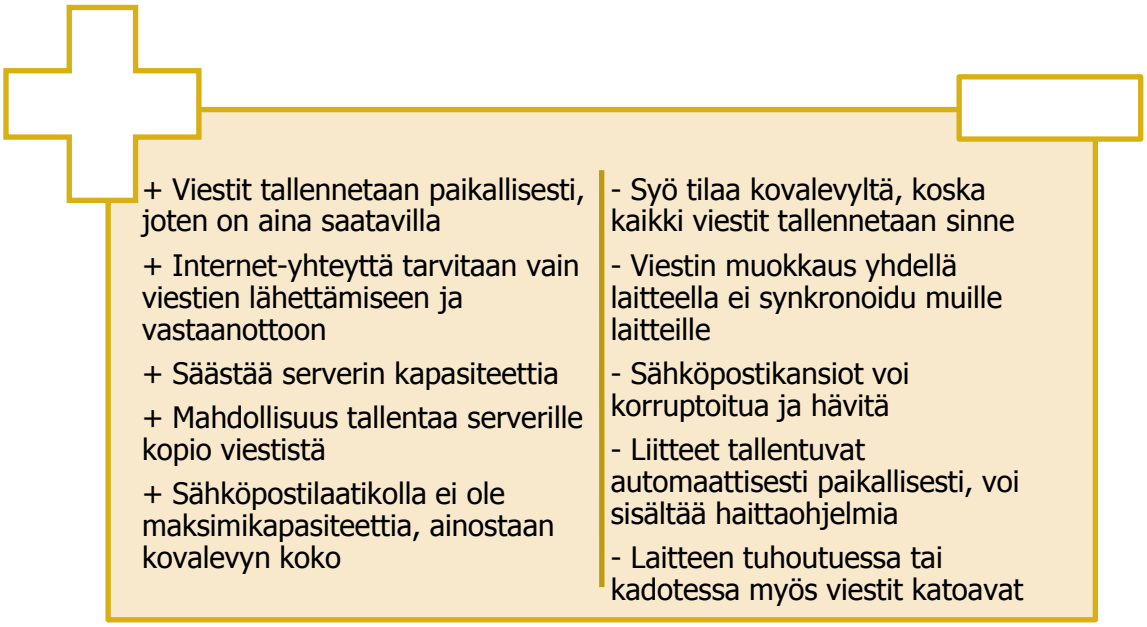
5.1.2 POP

POP on protokollamainen sovelluskehys sähköpostin hakuun postipalvelimelta. Käytetyin versio POP-protokollasta on POP3. POP3-protokolla sopii parhaiten käytettäväksi kun sähköpostia käytetään aina samalla työasemalla. (Dean 2009, 502.)

POP3-protokollan toimintaperiaate menee seuraavasti: 1. sähköpostiviestit säilötään postipalvelimella kunnes käyttäjä ottaa yhteyden postipalvelimelle 2. sähköpostiviestit ladataan paikalliselle koneelle ja poistetaan yleensä automaattisesti postipalvelimelta. POP3-protokolla säästää serverin resursseja, koska viestit poistetaan sieltä vastaanoton jälkeen. Lähes kaikki postipalvelimet ja clientit tukevat POP3-protokollaa. (Dean 2009, 502.)



KUVIO 6. IMAP-protokollan hyödyt ja haitat. (Tran 2018.)



KUVIO 7. POP3-protokollan hyödyt ja haitat. (Tran 2018.)

Päätelmänä voidaan sanoa IMAP-protokollan soveltuvan paremmin tämän tikettijärjestelmän sähköpostiviestien lukemista varten. IMAP soveltuu paremmin tikettijärjestelmään, koska yhtä tikettiä voi hallinnoida useampi eri henkilö, jolloin on tärkeää pitää tiketit synkronoituina. IMAP on myös turvallisempi ja nopeampi käyttää.

5.2 OWASP

OWASP on voittoa tavoittelematon organisaatio, joka keskittyy kehittämään sovellusten turvallisuutta (owasp.org). Alla oleva kuvio esittää OWASP:in määrittelemät kymmenen yleisintä turvallisuusriskiä vuodelta 2017. (Kuvio 8).

OWASP kymmenen yleisintä turvallisuusriskiä vuonna 2017	
1. Injektio	Hyökkääjän koodi voi huijata tulkkia suorittamaan ei-toivottuja komentoja tai pääsemään käsiksi dataan ilman kunnollisia valtuuksia.
2. Rikkinäinen autentikointi	Heikosti toteutetun autentikoinnin avulla hyökkääjä voi päästä käsiksi istuntoihin tai salasanoihin.
3. Arkaluontoisen datan heikko suojaus	Monet verkkosivut ja rajapinnat eivät suojaa arkaluotoista tietoa tarpeeksi hyvin. Hyökkääjät voi varastaa heikosti suojattua dataa ja syyllistyä identiteettivarkauteen, luottokorttivarkauteen tai muihin rikoksiin.
4. Ulkoiset viitteet XML-tiedostoissa	Ulkoisia viitteitä voidaan käyttää paljastamaan sisäisiä tiedostoja hyödyntäen sisäistä tiedostojakoa tai koodin etäsuorittamista.
5. Heikko oikeuksienhallinta	Rajoitukset käyttäjän tekemisiä koskien ovat monesti puutteellisesti määritelty tai niitä ei valvota ollenkaan. Hyökkääjä voi hyödyntää näitä heikkouksia päästäkseen käsiksi muiden käyttäjien tileihin tai arkaluonteisiin tiedostoihin.
6. Tietoturvan virheelliset määritykset	Tietoturvan virheellinen määritys on yleisin turvallisuusriski. Yleisiä virheitä ovat virheilmoitukset, jotka paljastavat arkaluonteista informaatiota, avoin pilvivarasto, väärin määritellyt http-headerit ja virheelliset yleismääritykset.
7. XSS-hyökkäykset	Hyökkääjä syöttää haitallista koodia esimerkiksi lomakkeen kautta ja voi täten kaapata käyttäjän istunnon, ohjata haitalliselle sivustolle tai vahingoittaa sivustoa. Syötteen validointi on puutteellista tai kokonaan olematonta.
8. Turvaton deserialisointi	Epävarma deserialisointi johtaa usein koodin etäsuorittamiseen. Vaikkei koodia etäsuoritettaisi, voidaan sitä käyttää hyökkäysten toteuttamiseen suorittamalla toistuvia hyökkäyksiä, injektioita tai oikeuksia vaativia toimenpiteitä.
9. Tunnettuja virheitä sisältävien komponenttien käyttö	Ulkoiset kirjastot, ohjelmistokehykset ja muut sovellusmoduulit käyttävät samoja oikeuksia kuin sovellus. Haavoittuvaista komponenttia hyödyntämällä hyökkääjä voi esimerkiksi päästä käsiksi tärkeisiin tietoihin.
10. Riittämätön kirjaaminen ja valvonta	Riittämätön kirjaaminen ja valvonta yhdistettynä puuttuvaan tai tehottomaan integraation vastaukseen sallii hyökkääjän päästä syvemmälle järjestelmään, ylläpitää pysyvyyttä, peukaloida, poistaa tai tuhota tietoja. Useat murrostutkimukset osoittavat, että murroksen havaitsemiseen kuluu yli 200 päivää. Tyypillisesti murros havaitaan ennemmin ulkopuolisilta tahoilta kuin sisäisestä prosessista tai seurannasta.

KUVIO 8. (OPEN WEB APPLICATION SECURITY PROJECT)

5.2.1 SQL-injektio

Sql-injektiolla tarkoitetaan hyökkäystä, jossa hyökätään järjestelmään syöttämällä käyttäjän syötteeseen sql-kysely. Onnistunut sql-injektio pääsee käsiksi tai suorittaa käyttöoikeuksia vaativia toimenpiteitä tietokantaan. SQL-injektiot voidaan jakaa kolmeen eri kategoriaan. Kanavansisäisessä sql-injektiossa data puretaan käyttämällä samaa kanavaa, jota käytetään SQL-koodin injektointiin. Kanavan ulkopuolisessa sql-injektiossa tietoja haetaan käyttämällä eri kanavaa. Sokealla sql-injektiolla tarkoitetaan hyökkäystä jossa ei ole varsinaista tiedonsiirtoa mutta testaaja pystyy muodostamaan tietoja lähettämällä erilaisia pyyntöjä ja hahmottamaan tietokantapalvelimen käyttäytymistä. (OPEN WEB APPLICATION SECURITY PROJECT.)

Onnistunut SQL-hyökkäys vaatii hyökkääjää hahmottamaan sql-kyselyn oikean syntaksin. Virheilmoituksissa kannatta huomioida mitä virheilmoituksessa kerrotaan. Kertomalla virheilmoituksessa sql-kyselyn syntaksin on hyökkääjän helppo ratkaista sql-kyselyn logiikka. SQL-hyökkäykset hyödyntävät yleisimmin kahden kyselyn yhdistämistä yhdeksi, boolean-arvoa, virheilmoituksia, ulkoisia kanavia sekä lisäämällä vastauksiin viivettä. (OPEN WEB APPLICATION SECURITY PROJECT.)

5.2.2 XSS

Cross-site scripting (XSS) on web-sovellusten yleisin turvallisuusuhka. Hyökkääjät hyödyntävät kyseistä uhkaa käyttäjäistunnon kaappaamiseen, ohjaamalla käyttäjän haitalliselle sivulle, seuraamalla näppäinkomentoja, manipuloimalla sivun sisältöä, lähettämällä dataa palvelimelle tai asentamalla haitallisen ohjelman käyttäjän koneelle. (Karande 2017, 3.)

6 TYÖN TOTEUTUS

6.1 Suunnittelu

Työn suunnittelu alkoi tikettijärjestelmän tavoitteiden määrittelyllä. Toivottuja toimintoja ja ominaisuuksia löytyi pitkä lista ja niistä valittiin opinnäytetyötä varten keskeiset asiat. Opinnäytetyön vaatimuksina tikettijärjestelmän täytyi pystyä lukemaan sähköpostilaatikkoa ja muodostamaan sähköposteista tikettejä, tarkistaa onko tiketti uusi vai vastaus olemassa olevaan ja liittämään vastaus tikettiin. Sähköpostilaatikon sähköpostit poistetaan tiketin tai kommentin luomisen jälkeen, koska tiketit ja kommentit tallentuvat tietokantaan. Tikettijärjestelmään haluttiin myös mahdollisuus vastata tikettiin ja mahdollisuus sulkea tiketti.

Koodin suunnittelussa täytyi perehtyä käytettäviin tekniikoihin ja vertailla niiden soveltuvuutta tikettijärjestelmään. Sähköpostien hakuun käytettävä protokolla täytyi selvittää vertailemalla IMAP:in ja POP3:n väliltä.

Koodaus tapahtui PHP-ohjelmointikielellä käyttäen Laravel-ohjelmointikehystä. Versionhallinta tapahtui Git-sovelluksen avulla ja koodit säilöttiin GitLabiin. Laravelin pakettien latausta helpottamaan käytettiin riippuvuuksienhallintatyökalu Composeria. Pakettien latauksessa hyödynnettiin pakettien keskeistä säilytyspaikkaa Packagistia.

Tietokantapalvelimena käytettiin MariaDB:tä, ja sen hallinnointia varten käytettiin HeidiSQL-sovellusta. Tietokantakyselyissä hyödynnettiin Laravelin Eloquentia. Artisania käytettiin apuna koodin suorittamisessa komentoriviltä. Sähköpostiviestien automaattista ajastettua hakua varten haluttiin käyttää Laravelin Jobia joka määriteltiin ajettavaksi tietyin väliajoin.

6.2 Toteutusvaiheet

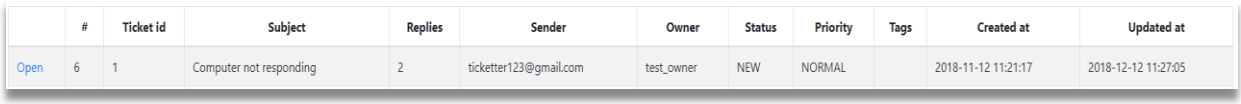
Itse koodin hahmottelu käynnistyi proof-of-concept-mallin avulla eli mahdollisimman yksinkertaisella versiolla, jossa testataan, että kaikki tarvittava saadaan toimimaan.

Sähköpostiviestien lukeminen IMAP- tai POP3-protokollaa käyttäen ei ollut entuudestaan tuttua, joten siihen täytyi perehtyä ensimmäisenä. Proof-of-concept-mallia varten luotiin sähköpostiosoite gmailiin. Yksinkertaisessa kirjautumislomakkeessa kysyttiin sähköpostiosoite ja salasana. Kirjautumisen yhteydessä kutsuttiin funktiota, joka ottaa yhteyden gmailin sähköpostilaatikkoon ja hakee sieltä sähköpostiviestit. Alla on kuvakaappaus (kuvio 9) yksinkertaisesta kirjautumislomakkeesta.

KUVIO 9. Demo-version kirjautumislomake.

Sähköposteista tutkittiin onko otsikossa tikettitunnusta. Uusi tiketti muodostuu, jos otsikosta ei löydy tikettitunnusta, mutta mikäli tikettitunnus löytyy, luodaan tiketille uusi kommentti. Tässä vaiheessa tikettitunnisteesta käytettiin muotoa ”risuaita + tiketin id” eli esimerkiksi #841.

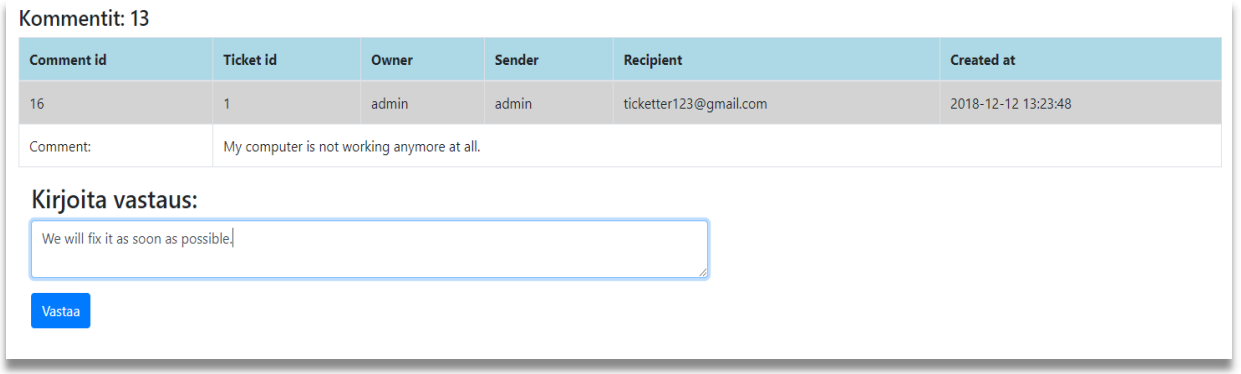
Sähköposteista luodut tiketit ja kommentit tallennettiin tietokantaan, jonka jälkeen sähköpostilaatikko tyhjennettiin IMAP:in delete ja expunge-funktioita hyödyntäen. Kirjautumisen jälkeen tiketit ja kommentit haetaan tietokannasta ja avataan taulukkonäkymä kaikista tiketeistä, jossa näkyy tikettien yleistiedot. (Kuvio 10). Oleellisia tietoja ovat muun muassa tiketin aihe, tila, lähettäjä ja tärkeys.



	#	Ticket id	Subject	Replies	Sender	Owner	Status	Priority	Tags	Created at	Updated at
Open	6	1	Computer not responding	2	ticketter123@gmail.com	test_owner	NEW	NORMAL		2018-11-12 11:21:17	2018-12-12 11:27:05

KUVIO 10. Tikettien listausnäkymä demo-versiossa.

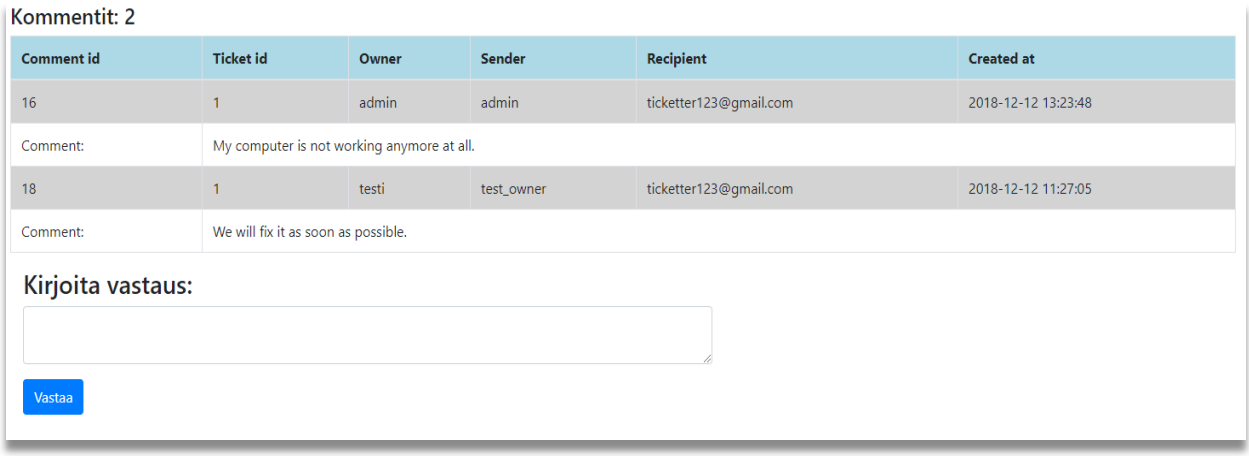
Klikkaamalla tiketin auki avautuu uusi sivu näyttäen tiketin tiedot ja tiketin vastaukset aikajärjestyksessä. Viimeisen kommentin jälkeen on tekstikenttä, johon voi kirjoittaa uuden kommentin. (Kuvio 11).



KUVIO 11. Yksittäisen tiketin näkymä demo-versiossa.

Lisäämällä uuden kommentin lähetetään sähköpostiviesti (kuvio 12 & kuvio 16), joka sisältää koko viestiketjun uusin kommentti ensimmäisenä. Näin on luotu yksinkertainen tikettijärjestelmä, jonka

avulla käyttäjä voi kommunikoida asiakkaan kanssa. Samalla uusi kommentti tallennetaan tietokantaan kyseisen tiketin kommentiksi.



KUVIO 12. Yksittäisen tiketin näkymä demo-versiossa.

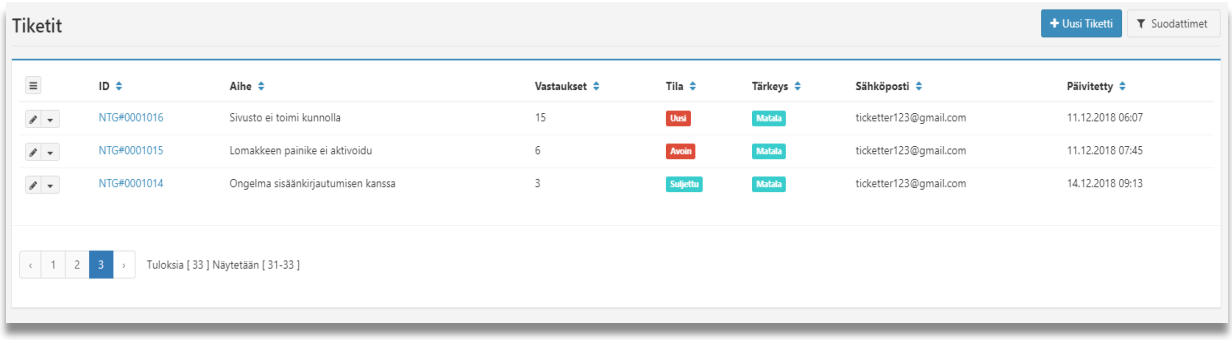
Tiketin luomisen varmistamiseksi luotiin testit, joilla varmistetaan, että koodi osaa eritellä tiketin ja kommentin. Tietokantaan kovakoodattiin kaksi tikettiä ja yksi vastaus, jotka kulkivat sovelluksen koodin läpi, joka tarkistaa onko kyseessä tiketti vai kommentti. Tarkistuksen yhteydessä viestit tallennetaan testi-tietokantaan kommentti- tai tikettitauluun. Testi suorittaa lopuksi tietokantakyselyn, jossa haetaan lukumäärät tiketti ja kommenttitaulusta. Koska kovakoodattuja tikettejä on kaksi ja kommentteja yksi, täytyy myös tietokannasta löytyä vastaavat määrät.

Lopullisen version tuli olla NettiTiedon toiminnanohjausjärjestelmään kuuluva paketti, jota varten oli omat säännökset. Lopullisen version koodit olivat melko pitkälti jo valmiina proof-of-concept-mallissa. Koodin rakennetta täytyi muokata yhteensopivaksi toiminnanohjausjärjestelmän koodin kanssa, mikä vei jonkin verran aikaa. Ulkoasu muuttui selkeästi, koska lopullisen version täytyi mukailla toiminnanohjausjärjestelmän ulkoasua. Lopulliseen versioon lisättiin vielä mahdollisuudet sulkea tiketti ja avata suljettu tiketti. Koodit säilöttiin GitLabissa ja sitä siirrettiin sinne sitä mukaa kun uutta koodia kirjoitettiin. Näin ollen versionhallinta pysyi selkeänä.

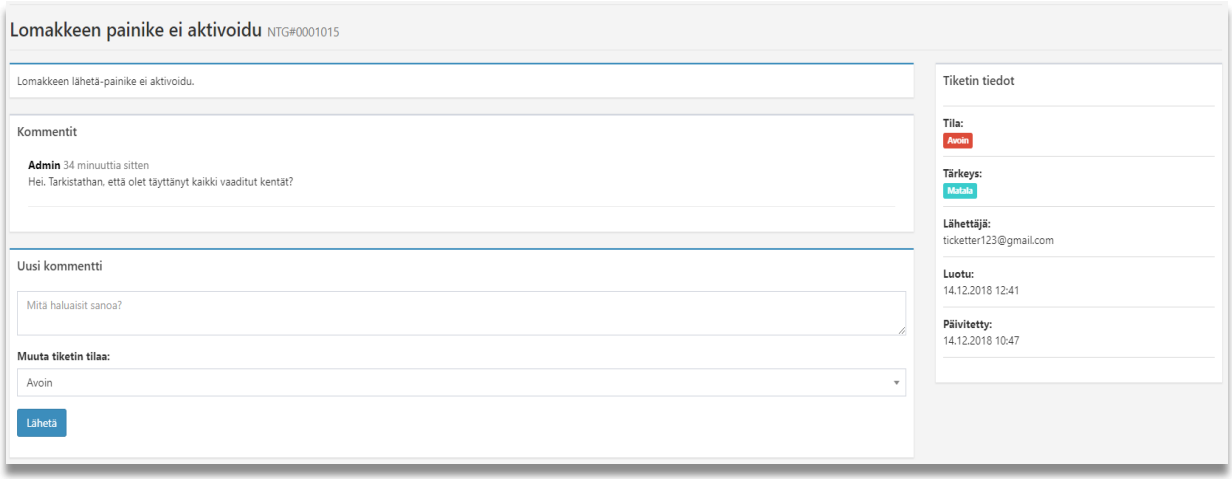
6.3 Lopputulos

Lopputuloksena syntyi Ticketer-niminen Laravel-paketti osaksi NettiTiedon toiminnanohjausjärjestelmää, jonka avulla voidaan käydä keskustelua asiakkaan ja yrityksen työntekijän välillä. Ticketerin ulkoasu sulautuu osaksi toiminnanohjausjärjestelmän ulkoasua. Ticketer-paketti hakee tietystä sähköpostiosoitteesta viestit ja muodostaa niistä tiketin tai kommentin. Ticketer listaa tiketit siistiksi näkymäksi, josta voi aukaista minkä tahansa tiketin ja tarkastella tiketin kommentteja tai vastata tikettiin/kommenttiin.

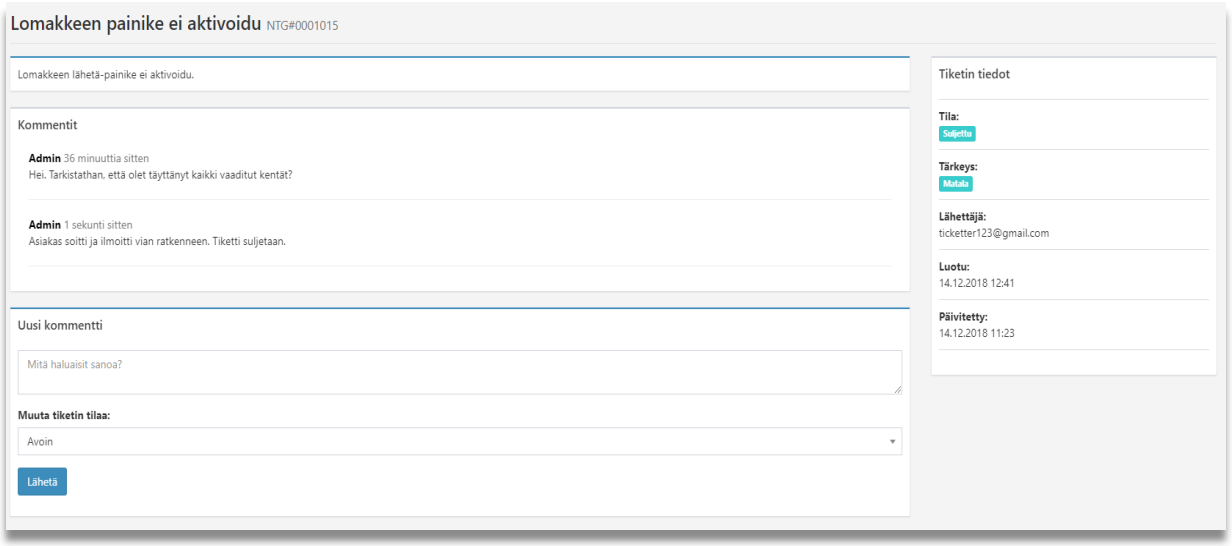
Jokainen sähköpostiviestistä muodostettu tiketti saa aluksi tilan ”uusi” (Kuvio 13). Kommentoimalla tikettiä on mahdollista muuttaa tiketin tilaa muuttamalla se avoimeksi tai suljetuksi (Kuvio 14 & Kuvio 15). Suljetun tiketin voi myös aukaista uudestaan, mikäli sellainen tarve esiintyy.



KUVIO 13. Tikettien listaus-näkymä.



KUVIO 14. Avoin tiketti, johon on vastattu.



KUVIO 15. Suljettu tiketti.

KUVIO 16. Tiketin kommentti lähetetty sähköpostiin.

7 POHDINTA

Opinnäytetyön aiheen kuultuani mietin hetken, että mitä tiketillä tarkoitetaan. Kuitenkin toimeksiantajan kertoessa työn tarpeista ja vaatimuksista alkoi käsite tiketti selkeytymään. Viimeistään tutustuessani itsenäisesti erilaisiin tikettijärjestelmiin ymmärsin täysin mitä opinnäytetyöni tulisi käsittelemään.

Opinnäytetyöni tehtyäni minun on vaikea kuvitella työskenteleväni yrityksessä, joka ei käyttäisi jotain tikettijärjestelmää. Sen hyödyt ovat kuitenkin niin suuret ja erilaisia tikettijärjestelmiä on saatavilla ihan ilmaiseksi.

Lopputulos auttoi ymmärtämään tikettijärjestelmän merkityksen yrityksen käytössä. Tikettijärjestelmä on tarpeellinen ainakin yrityksille, jotka saavat paljon tukipyyntöjä asiakkailta. Onko tikettijärjestelmälle käyttöä yksityisyrittäjällä tai muutaman työntekijän kokoisella yrityksellä? Mielestäni on, mutta toimeen voi tulla ilmankin. Tikettijärjestelmä selkeyttää tukipyyntöjen hallintaa kun kaikki pyynnöt ovat tallessa selkeästi järjestelmässä eikä siellä täällä pitkin sähköpostilaatikkoa ja ne voidaan hoitaa selkeässä järjestyksessä. Mikäli yritykselle ei kuitenkaan tule tukipyyntöjä viikossa muutamaa enempää voidaan ne hoitaa myös ilman tikettijärjestelmää varsin pienellä vaivalla.

Tikettijärjestelmän suurimmat hyödyt tulevat tukipyyntöjen selkeästä hallinnasta. Mikäli avoimia tikettejä on useita, voidaan ne järjestää jonoon ja käsitellä tärkeysjärjestyksessä. Automaattinen ajastettu sähköpostilaatikon lukeminen vähentää käsin tehtävän työn määrää. Automaattisen tietokantaan tallentamisen jälkeen sähköpostilaatikosta poistetaan luetut viestit, jolloin tätäkään ei tarvitse tehdä käsin.

Opinnäytetyön toteutus tapahtui NettiTiedon toimistolla Volttikadulla, joka oli mielestäni erittäin onnistunut ratkaisu. Työvälineet olivat kunnossa ja sain apua aina sitä pyytäessäni. Näin opinnäytetyö eteni sujuvasti päivä päivältä. Arvioitu aikataulu tiukentui alkuperäisestä suunnitelmasta kun suunniteltu 8 tuntia päivässä muuttui noin 6 tunniksi päivää kohden. Tämä johtui osa-aikaisista iltatöistäni toisessa yrityksessä.

Lopputulos pystyi lukemaan sähköpostilaatikosta viestit ajastetusti ja päättelemään onko kyseessä uusi tiketti vai kommentti tikettiin. Kommentoimalla tikettiä luodaan uusi kommentti joka myös lähetetään sähköpostilla tiketin lähettäjälle. Näin ollen sovelluksella pystyi käymään keskustelua asiakkaan ja työntekijän välillä. Opinnäytetyön lopullinen versio täytti sille asetetut vaatimukset.

8 JATKOKEHITYS

Tikettijärjestelmän jatkokehitystä ajatellen siitä puuttuu vielä paljon toimintoja. Nykyinen versio muodostaa tiketin ainoastaan sähköpostiin tulleesta viestistä. Tiketti pitäisi pystyä luomaan myös järjestelmästä käsin tai erilaisten yhteydenottolomakkeiden kautta.

Sähköpostiin tulleista viesteistä täytyy pystyä jatkossa karsimaan roskapostiviestit ja haitalliset viestit pois. Sähköpostiviestien lukemiseen täytyy lisätä tarkistus, jottei käyttäjän syöttämää haitallista koodia päädy tietokantaan. Täytyy myös huomioida ettei luokitella oikeita viestejä roskaposteiksi.

Tiketin vastauksena lähetettävää sähköpostiviestiä täytyy muokata tyylikkäämmäksi. Tämänhetkinen viesti on tyylittelemätön, koska sen muotoilulle ei jäänyt juurikaan aikaa. Muita mahdollisia jatkokehityskohteita ovat tikettien jakaminen ja yhdistäminen, jonojen lisäys, avainsanojen lisäys ja FAQ-vastaukset toistuvasti esiintyville ongelmille.

LÄHTEET

BHAGAT, Varum. 2017. Features That Make PHP Laravel Framework Best [blogi]. [Viitattu 2018-11-11.] Saatavissa: https://www.pixelcrayons.com/blog/web/8-top-features-stats-facts-about-php-laravel-framework/

CHACON, Scott ja STRAUB, Ben. 2009. Pro Git 1st Edition. 2 Gitin perusteet – Muutosten tallennus tietolähteeseen. [Viitattu 2018-10-15.] Saatavissa: https://git-scm.com/book/fi/v1/Gitin-perusteet-Muutosten-tallennus-tietol%C3%A4hteeseen

CHACON, Scott ja STRAUB, Ben. 2009. Pro Git 1st Edition. [Viitattu 2018-10-15.] Saatavissa: https://git-scm.com/book/fi/v1/Alkusanat-Gitin-perusteet

CHACON, Scott ja STRAUB, Ben. 2014. Pro Git 2nd Edition. [Viitattu 2018-10-15.] Saatavissa: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

DEAN, Tamara. 2009. Network+ Guide to Networks [e-kirja]. [Viitattu 2018-10-16.] Saatavissa: https://books.google.fi/books?id=UD0h_GqgbHgC&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

GETCOMPOSER. Basic usage [verkkojulkaisu]. [Viitattu 2018-10-02.] Saatavissa: https://getcomposer.org/doc/00-intro.md

GETCOMPOSER. Introduction [verkkojulkaisu]. [Viitattu 2018-10-02.] Saatavissa: https://getcomposer.org/doc/01-basic-usage.md

GISUSER. 2018. A Brief History of ITIL [verkkojulkaisu]. [Viitattu 2018-11-14.] Saatavissa: http://gisuser.com/2018/03/a-brief-history-of-itil/

GREYCAMPUS. What is ITIL? [verkkojulkaisu]. [Viitattu 2018-11-14.] Saatavissa: https://www.greycampus.com/opencampus/itil-foundation/what-is-itil

HEIDISQL. What's this? [verkkojulkaisu]. [Viitattu 2018-11-05.] Saatavissa: https://www.heidisql.com/

IRWIN, Luke. 2018. The 5 stages of ITIL [blogi]. IT Governance Blog. [Viitattu 2018-11-14.] Saatavissa: https://www.itgovernance.co.uk/blog/the-5-stages-of-itil

KARANDE, Chetan. 2017. Securing Node Applications, 1st Edition [e-kirja]. O'Reilly Media, Inc. [Viitattu: 2018-11-05.] Saatavissa: https://proquest-safaribooksonline-com.ezproxy.savonia.fi/9781491982426

KONONENKO, Kevin. 2018. Front End v. Back End Explained by Waiting Tables at a Restaurant [blogi]. [Viitattu 2018-12-03.] Saatavissa: https://blog.codeanalogies.com/2018/04/07/front-end-v-back-end-explained-by-waiting-tables-at-a-restaurant/

KOLODIY, Sergey. Unit Test, How to Write Testable Code and Why It Matters [blogi]. [Viitattu-2018-11-15.] Saatavissa: https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters

MARIADB. About MariaDB [verkkojulkaisu]. [Viitattu 2018-11-05.] Saatavissa: https://mariadb.org/about/

MCCOOL, Shawn. 2012 Laravel Starter [e-kirja]. Birmingham: Packt Publishing Ltd. Saatavissa: http://web.b.ebscohost.com.ezproxy.savonia.fi/ehost/ebookviewer/ebook/bmxlYmtfXzYxNzgwN19fQU41?sid=f5344c13-cd1b-43b7-8af2-0aedcadbabe0@pdc-v-sessmgr02&vid=0&format=EB&rid=1

MICROSOFT OFFICE. Perustiedot tietokannasta [verkkoaineisto]. [Viitattu-2018-11-15.] Saatavissa: https://support.office.com/fi-fi/article/perustiedot-tietokannasta-a849ac16-07c7-4a31-9948-3c8c94a7c204

NETTITIETO OY. NettiTieto [verkkoaineisto]. [Viitattu 2018-11-15.] Saatavissa: https://www.net-titieto.fi/

O'BRIEN, Jesse. A Brief History of Laravel [blogi]. [Viitattu 2018-11-16.] Saatavissa: https://me-dium.com/vehikl-news/a-brief-history-of-laravel-5d55970885bc

OPEN WEB APPLICATION SECURITY PROJECT (OWASP): Testing for SQL Injection [verkkojulkaisu]. [Viitattu 2018-10-30.] Saatavissa: https://www.owasp.org/index.php/Testing_for_SQL_Injec-tion_(OTG-INPVAL-005)

OPEN WEB APPLICATION SECURITY PROJECT (OWASP): OWASP Top-10 – 2017 [verkkojulkaisu]. [Viitattu 2018-11-21.] Saatavissa: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

PACKAGIST. Statistics: Packages/versions over time [verkkojulkaisu]. [Viitattu 2018-11-06.] Saata-vissa: https://packagist.org/statistics

PHP.net. What Is PHP? [Verkkojulkaisu.] [Viitattu 2018-10-15.] Saatavissa: http://php.net/ma-nual/en/intro-whatis.php

IGHODARO, Neo. 2018. How Laravel Implements MVC and How to Use it effectively [blogi]. [Viitattu 2018-12-03.] Saatavissa: https://blog.pusher.com/laravel-mvc-use/

SANTACROCE, Ferdinando. 2015. Git Essentials [e-kirja]. Birmingham, UK: Packt Publishing. Saa-tavissa: http://web.a.ebscohost.com.ezproxy.savonia.fi/ehost/ebookviewer/ebook/bmx-lYmtfXzk4NjM2Nl9fQU41?sid=5ef7e635-b287-4249-837a-172622ee06a2@ses-sionmgr4007&vid=0&format=EB&rid=1

SAUNIER, Raphael. 2014. Getting Started with Laravel 4 [e-kirja]. [Viitattu 2018-10-09.] Saatavissa: http://web.a.ebscohost.com.ezproxy.savonia.fi/ehost/ebookvie-wer/ebook/bmxlYmtfXzY4OTQwNF9fQU41?sid=9f511900-daaf-4ae9-8eb6-b912534c10cf@session-mgr4006&vid=0&format=EB&rid=1

SCOTT, Adam. 2017. Collaborative Web Development 2017 [e-kirja]. [Viitattu 2018-12-03.] Saata-vissa: https://proquest-safaribooksonline-com.ezproxy.savonia.fi/book/web-develop-ment/9781491996096/1dot-introduction/idm139955987348320_html#X2ludGVy-bmFsX0h0bWxWaWV3P3htbGlkPTk3ODE0OTE5OTYwOTYlMkZpZG0xMzk5NTU5ODcxNDExMDR-faHRtbCZxdWVyeT0=

SPIER,R. , VINCENT,J. , FOLEY,R. , ROLSKY,D. ja CHAMBERLAIN,D. 2008. RT Essentials. A Dis-sected Ticketing System [e-kirja]. [Viitattu 2018-10-02.] Saatavissa: https://www.oreilly.com/li-brary/view/rt-essentials/0596006683/ch01.html

SPIER,R. , VINCENT,J. , FOLEY,R. , ROLSKY,D. ja CHAMBERLAIN,D. 2008. RT Essentials. Customer Service [e-kirja]. [Viitattu 2018-10-02.] Saatavissa: https://www.oreilly.com/library/view/rt-essen-tials/0596006683/ch01.html

SURGUY, Maksim 2013-07-27. History of Laravel PHP framework, Eloquence emerging [blogi]. [Vii-tattu 2018-10-09.] Saatavissa: https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/

TAMPEREEN TEKNILLINEN YLIOPISTO: Tietoturva [verkkojulkaisu]. [Viitattu 2018-10-03.] Saatavissa: http://www.cs.tut.fi/~seitti/2015/kalvot/tietoturva/all.html

THE TECH TERMS COMPUTER DICTIONARY. Markup Language [verkkojulkaisu]. [Viitattu 2018-11-21.] Saatavissa: https://techterms.com/definition/markup_language

TRAN, Phuong. 2018. Email Protocol: IMAP vs. POP3 [verkkojulkaisu]. [Viitattu 2018-11-15.] Saatavissa: https://paubox.zendesk.com/hc/en-us/articles/115005758567-Email-Protocol-IMAP-vs-POP3

W3SCHOOLS. PHP 5 Introduction [verkkojulkaisu]. [Viitattu 2018-11-21.] Saatavissa: https://www.w3schools.com/php/php_intro.asp

WHITE, Sarah ja GREINER, Lynn. 2017. What is ITIL? Your guide to the Infrastructure Library [verkkojulkaisu]. [Viitattu 2018-11-14.] Saatavissa: https://www.cio.com/article/2439501/itil/infrastructure-it-infrastructure-library-itil-definition-and-solutions.html

WODEHOUSE, Carey. Web Development Languages 101 [verkkojulkaisu]. [Viitattu 2018-12-03.] Saatavissa: https://www.upwork.com/hiring/development/web-development-languages-101/