SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

QAIRA

Regulaatioiden seuranta- ja raportointityökalu lääkinnällisten laitteiden valmistajille

TEKIJÄ: Lauri Heinonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Lauri Heinonen	
Työn nimi QAiRA – regulaatioiden seuranta- ja raportointityökalu lääkinnällisten laitteiden valmistajille	
Päiväys 11.11.2021	Sivumäärä/Liitteet 40
Ohjaaja(t) Lehtori, Mika Vanhanen, Lehtori, Keijo Kuosmanen	
Toimeksiantaja/Yhteistyökumppani(t) Kasve Oy	

Tiivistelmä

Opinnäytetyön aiheena oli Kasve Oy:n tuottaman QAiRA-verkkosovelluksen uusien ominaisuuksien ja toiminnallisuuksien toteutus osana sovelluskehitysprosessia. Yhtenä pääasiallisena kehityskohteena oli kahden eri asiakastietolähteen – sovelluksen tietokannan sekä yrityksen käyttämän Severa-järjestelmän – tietojen synkronointi niin kutsutulla asiakastietointegraatiolla.

Sovellukseen suunniteltiin ja toteutettiin opinnäytetyön puitteissa kolme erillistä, mutta toisiinsa osin limittyvää kokonaisuutta. Ensimmäinen kokonaisuus laajensi QAiRAn olemassa olevia regulaatioiden seurantatyökaluja koostamalla regulaatioiden sisältöpäivityksessä syntyvät muutostiedot QAiRAn viikottaiseen sisältöpäivityskoosteviestiin lisäten käyttäjille myös käyttöliittymäkomponentit uusien viestiasetusten hallinnointiin. Toinen kokonaisuus koostui järjestelmäylläpidolle toteutetusta sovelluksen tuottamien automatisoitujen sähköpostiviestien lähetystyökalusta. Kolmas kokonaisuus piti sisällään aiemmin mainitun integraation projektin kahden asiakastietolähteen välillä. Opinnäytetyö koostui teknisen toteutuksen suunnittelusta ja implementoinnista järjestelmään. Toteutettujen ominaisuuksien frontend-toiminnallisuudet kirjoitettiin JavaScript-kielellä käyttäen jQuery-kirjastoja. Backend-prosessointi toteutettiin PHP:llä käyttäen MariaDB-tietokantaa. Sähköpostien automatisoituun lähetykseen käytettiin kolmannen osapuolen Mailgun-palvelua ja asiakastietointegraation datan synkronointi toteutettiin HTTP-pyynnöin Visman kehittämän Severa-järjestelmän REST-rajapinnan kautta.

Opinnäytetyön tuloksina syntyneet toiminnallisuudet julkaistiin sovelluksen tuotantoversioon osana kehitysprojektin normaalia julkaisuprosessia. Suunnitelman mukaisen asiakastietointegraation määrittely ja toteutus allokoitiin projektinhallinnollisista syistä muiden toteutettujen toiminnallisuuksien kustannuksella aiottua myöhemmälle, mistä johtuen sen osuus opinnäytetyössä jäi aiottua pienemmäksi ja jätti varmasti tilaa myös jatkokehitykselle. Kaikki toteutetut toiminnallisuudet koettiin kuitenkin erittäin tarpeellisiksi ja niiden tuottama hyöty jakautui monipuolisesti eri käyttäjäryhmien kesken sovelluksen loppukäyttäjistä järjestelmäylläpitäjiin ja yrityksen sisäisten markkinointi- ja liiketoiminnan hallinto-osastojen käyttäjiin.

Avainsanat
PHP, rajapinnat, verkko-ohjelmointi, käyttöliittymät

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Lauri Heinonen			
Title of Thesis QAiRA – Regulation Monitoring and Reporting Tool for Medical Device Manufacturers			
Date	11 November 2021	Pages/Appendices	40
Supervisor(s) Mr. Mika Vanhanen, Senior Lecturer and Mr. Keijo Kuosmanen, Senior Lecturer			
Client Organisation /Partners Kasve Ltd			

Abstract

The purpose of this thesis was to develop new features and functionalities to Kasve Ltd's QAiRA web application as part of regular web development process. One main development objective was to create a so-called customer data integration by synchronizing the data between two separate data sources – the application database and the Severa system used by the company.

Three separate but partly overlapping bodies of work were designed and implemented to the application as part of the thesis project. The first subfeature expanded on QAiRA's existing regulatory follow-up tools by compiling the regulation content update data into QAiRA's weekly content update notifications message and by adding the user interface components to manage the new message settings. The second feature consisted of an internal tool created for the system administrators for sending automated emails generated by the application. The third feature included the aforementioned integration between the project's two customer data sources. The thesis was composed of the planning and the technical implementation to the system. The frontend functionalities of the implemented features were written in JavaScript by utilizing the jQuery libraries. The backend processing was implemented with PHP using MariaDB database. The delivery of the automated emails was created by using a third-party service Mailgun and the data synchronization of the customer data integration was done with HTTP requests via Severa system's REST interface developed by Visma.

The features created during the thesis project were released to the production version as part of the normal release process of the development project. Due to reasons in internal project management, the specification and the implementation of the planned customer data integration were allocated to a later part of the project after the other implemented features which caused its portion in the thesis project to be diminished from the intended scope and surely left room also for future development. All of the implemented features were nonetheless received as important additions to the application and their benefits were diversely distributed between the application's different user groups from the end users to the system administrators and the users of the company's internal marketing and business management departments.

Keywords
PHP, interfaces, web development, user interfaces

ESIPUHE

Haluan kiittää Kasve Oy:tä opintojeni tukemisesta, sekä mahdollisuudesta toteuttaa opinnäytetyö osana kehitysprojektia. Yrityksen sisältä haluan erityisesti kiittää muuta kehitysryhmää; Vincent Goguetia, Teemu Litmasta, Valtteri Kovalaista ja Anniina Kopsaa. Haluan kiittää myös opinnäytetyön yhteyshenkilöinä toimineita Mikko Juutia ja Ville Heleniusta.

Lisäksi haluan kiittää Savonia-ammattikorkeakoulua koko opintopolkuni ajalta. Erityisesti haluan osoittaa kiitokseni opinto-ohjaajalleni Sami Lahdelle, jonka huolenpito opintojen sujumisesta on ollut korvaamatonta; opinnäytetyön ohjaajalleni Mika Vanhaselle, jolta olen saanut erittäin hyviä neuvoja koko opinnäytetyönprosessin ajan; opinnäytetyön toiselle ohjaajalle Keijo Kuosmaselle, joka tarjosi hyviä näkemyksiä opinnäytetyöprojektin suunnittelusta aloituspalaverissa; sekä kielten opettajille Teija Tossavaiselle ja Anna-Maija Pietilälle neuvoista, avusta ja kieliasun tarkistuksista.

Suurimmat kiitokset haluan osoittaa puolisolleni Mirvalle, joka on suuresti tukenut ja kannustanut minua opinnäytetyötä tehdessäni, sekä esikoisellemme, jonka hymy on sekä inspiroinut että voimaannuttanut keskeneräisten tehtävien suorittamisessa.

Kuopiossa 15.11.2021
Lauri Heinonen

SISÄLTÖ

1 JOHDANTO

Opinnäytetyön aiheena on kehittää uusia ominaisuuksia ja osatoiminnallisuuksia Kasve Oy:n tuottamaan QAiRA-verkkosovellukseen osana projektissa käytettyjä kehitys- ja dokumentointiprosesseja. Tarkoituksena on julkaista yksi tai useampi kehitetty toiminnallisuus sovelluksen tulevien versiopäivityksien yhteydessä. Yhtenä kehitettävien uusien ominaisuuksien painopisteenä on niin kutsutun asiakastietointegraation suunnittelu ja implementointi.

Opinnäytetyöntekijä on työskennellyt QAiRA-projektissa vuoden 2019 alusta sovelluksen prototyyppivaiheesta lähtien, joten hänellä on jo ennestään kokemusta projektin käytänteistä ja sovelluksen järjestelmästä. Tekijä toteuttaa kaikki kehitettävät ominaisuudet itsenäisesti. Muu kehitysryhmä on mukana ominaisuuksien lopullisessa testaamisessa ja päivittämisessä tuotantoympäristöön versiopäivityksen yhteydessä.

Ominaisuuksien kehitys toteutetaan verkkosovelluskehityksenä pääosin käyttäen projektissa käytössä olleita teknologioita. Sovelluksen frontend toteutetaan JavaScriptillä käyttäen jQuery-kirjastoja. Backend toteutetaan PHP:llä ja tietokantana toimii MariaDB.

Ohjelmointityö paikallisessa kehitysympäristössä tehdään Atom-editorilla käyttäen XAMPP-ohjelmistoa backend-prosessointiin. Toteutettavat ominaisuudet dokumentoidaan Confluence-järjestelmään ja ominaisuuksien työnkulkuvaiheet ja julkaisuaikataulut kirjataan käyttäen Jiraa. Versionhallintajärjestelmänä toimii Git ja tietovarasto on jaettu Gitlab-palvelun kautta.

Asiakastietointegraatiossa hyödynnetään Severa-järjestelmää integraation toisena datalähteenä. Tietojen siirto toteutetaan järjestelmän REST-rajapinnan kautta.

2 LYHENTEET JA MÄÄRITELMÄT

Ajax	Asynchronous JavaScript and XML; dynaamisille verkkosovelluksille tyypillinen asynkroninen tiedonsiirto-operaatio käyttäen JSON- tai XML-tiedonsiirtoformaatteja
API	Application Programming Interface; rajapinta, jonka kautta sovellus voi hakea ja kirjoittaa tietoja suoraan toisen sovelluksen järjestelmään
Asynkroninen	Prosessi, joka suoritetaan irtaallaan muusta ohjelmakoodin toiminnasta
Backend	Verkkosovelluskehityksessä palvelinkoneilla suoritettavan ohjelmakoodin ja säilytettävän tietokannan sisältävä ympäristö
CSV	Comma-separated values; tiedostomuoto, jossa joukko toisiinsa liittyviä yksittäisiä tietoarvoja on erotettu toisistaan pilkulla tai vastaavalla erotinmerkillä
Dynaaminen sovellus	Erityisesti verkkosovelluskehityksessä sovellus, joka muokkaa suoritettavan sivun rakennetta vain ladattavien tietojen osalta
Frontend	Verkkosovelluskehityksessä asiakasohjelman eli käytännössä verkkoselaimen tulkitsema ohjelmakoodi ja lataama data
Funktio	Ohjelmakoodin osa, joka suorittaa siihen ohjelmoituja toimintoja annettujen parametrien mukaan ja tarvittaessa palauttaa prosessoimaansa dataa ennaltamääritellyssä muodossa
HTTP	Hypertext Transfer Protocol; verkkotiedonsiirron mahdollistava protokolla, jossa tietoa siirretään ennaltamäärättyjen pyyntöjen ja vastausten kautta
Instanssi	Ennaltamääritettyyn malliin perustuva yksittäinen suorituskerta tai konteksti
Iterointi	Operaatio, jossa samaa prosessia toistetaan monta kertaa muuttuvin parametrein
JSON	JavaScript Object Notation; tiedonsiirtoformaatti, jossa tieto jäsennellään JavaScript-kielellä määritettyjen objektien muotoon

Kirjasto	Kokoelma yleisten toimintojen suorittamiseen kirjoitettua ohjelmakoodia (esimerkiksi funktioita ja luokkia), jota voidaan hyödyntää monipuolisesti eri käyttötarkoituksiin
Konfigurointi	Laitteen, sovelluksen tai ympäristön asetusten hallinnointi
Luokka	Ohjelmakoodin osa, joka mallintaa jotain konkreettista tai abstraktia kokonaisuutta muun muassa sille määritettyjen ominaisuuksien ja metodeiksi kutsuttujen funktioiden avulla
Objekti	Toisin sanoen olio; luokan määrittämän mallin mukaisesti luotu yksittäinen instanssi mallinnetusta kokonaisuudesta
Parametri	Esimerkiksi funktiolle syötettävä muuttuja eli tietoarvo, joka vaikuttaa funktion suorittamiin operaatioihin ja palauttamaan dataan
Refaktorointi	Ohjelmakoodin toiminnallisuutta muokkaamaton optimointi, yksinkertaistaminen tai luettavuuden parantaminen
REST	Representational state transfer; ohjelmistorakennemalli, jonka avulla eri sovellukset voivat rajapinnan kautta siirtää tietoja HTTP-pyyntöjen avulla
Skripti	Itsekseen suoritettavaksi tarkoitettu ohjelma, jonka esimerkiksi järjestelmäylläpitäjä voi käynnistää tai joka voidaan asettaa suorittamaan automaattisesti säännöllisin väliajoin
SMTP	Simple Mail Transfer Protocol; sähköpostiviestien lähetykseen ja vastaanottoon käytetty protokolla
SQL	Structured Query Language; pääosin tietokantajärjestelmän hallintaan ja tietokantakyselyihin käytetty ohjelmointikieli
Wiki	Yhteisönsä muokattavissa oleva hypertekstijulkaisu tai esimerkiksi dokumentaatio

3 TYÖN TILAAJA

KUVA 1 Opinnäytetyön tilaajan, Kasve Oy:n logo

Opinnäytetyön tilaaja, Kasve Oy on vuonna 2012 perustettu terveysalan asiantuntija- sekä koulutuspalveluita tarjoava suomalainen yritys. Opinnäytetyön toteutuksen aikana vuonna 2021 Kasve työllisti noin 15 henkilöä pääosin Kuopiossa. (Kasve.)

Kasven asiantuntijapalvelut on suunnattu erityisesti lääkinnällisten laitteiden ja ohjelmistojen valmistajille. Olennaisimmillaan asiantuntijapalvelut pitävät sisällään muun muassa kehitettävän laitteen tai ohjelmiston tuotteistukseen tarvittavaa regulaatiohallinnan ja -seurannan, sekä laatujohtamisen tai liiketoiminnan kehittämisen ohjausta. (Kasve.)

3.1 QAiRA

KUVA 2 Kasven tuottaman QAiRA-sovelluksen logo

QAiRA on Kasven kehittämä ja tuottama ohjelmistopalvelu, joka tarjoaa työkaluja regulaatioiden hallintaan ja seurantaan, laatuvaatimusten täyttämisen todentamiseen, sekä raportointiin. QAiRAn kautta Kasve pyrkii tuottamaan digitaalisia työkaluja tarjoamiensa asiantuntijapalveluiden rinnalle palvellakseen yhä suurempaa asiakaskuntaa. Sovelluksen nimi tulee laadunvarmennusta ja regulatiivisia asioita tarkoittavasta QA/RA-lyhenteestä, joka puolestaan tulee sanoista ”Quality Assurance” ja ”Regulatory Affairs”.

Jäsennellymmin lueteltuna QAiRAn nykyiset toiminnallisuudet voidaan jakaa kolmeen kokonaisuuteen: regulaatiohallintaan, uutissyötteiden seurantaan, sekä organisaatio- ja tuotekohtaisiin työkaluihin (Kasve). Opinnäytetyön kannalta edellä mainituista kokonaisuuksista olennaisin on regulaatiohallinta, jolla voidaan seurata erityisesti terveysalan lääkinnällisiä laitteita ja ohjelmistoja sääntelevää lainsäädäntöä, lainsäädäntöä tarkentavia ohjeistuksia, sekä lainsäädännön yhteneväisen toteutuksen mahdollistamiseksi laadittuja standardeja.

3.2 Lääkinnälliset laitteet, regulaatiot ja standardit

”Lääkinnällinen laite” on termi, jota käytetään kuvaamaan terveysalan laite- ja ohjelmistovalmistajien kehittämiä tuotteita. Lääkinnälliset laitteet voidaan jakaa kahteen eri ryhmään: lääkinnällisiin laitteisiin (lyhennettynä MD sanoista Medical Device) ja in vitro -diagnostiikkaan tarkoitettuihin lääkinnällisiin laitteisiin (lyhennettynä IVD sanoista In Vitro Diagnostics). Euroopan komission tuottaman lääkinnällisiä laitteita koskevan artikkelin mukaan lääkinnällisiksi laitteiksi ”luetaan esimerkiksi laastarit, piilolinssit, röntgenlaitteet, sydämentahdistimet, rintaimplantit, sovellukset ja keinotekoiset lonkkanivelet. In vitro -diagnostiikkaan tarkoitettuja lääkinnällisiä laitteita käytetään testien tekemiseen näytteistä. Niihin kuuluvat esimerkiksi HIV-verikokeet, raskaustestit ja diabeetikoille tarkoitetut verensokerin seurantajärjestelmät.” (Euroopan komissio.)

Euroopan Unionin alueella lääkinnällisten laitteiden toimialaa säännellään sääntelykehyksellä, joka pyrkii suojelemaan potilaiden ja lääkinnällisten laitteiden käyttäjien terveyttä, sekä yhtenäisen viitekehyksen tarjoten varmistamaan alan sisämarkkinoiden toimivuuden. Tieteen ja teknologian kehittyessä myös lainsäädännön on muututtava vastaamaan tuotettuja laitteita ja ohjelmistoja. Huomioarvoisesti opinnäytetyön toteutuksen aikana vuonna 2021 on ollut voimassa siirtymäkausi, jonka aikana sääntelykehyksen kolme direktiiviä korvataan kahdella uudella asetuksella. (Euroopan komissio.)

Myös standardien on alati päivityttävä vastatakseen muuttuvaan lainsäädäntöön. Standardeja julkaisevat itsenäiset standardointijärjestöt. Standardointijärjestöt jakautuvat laajuutensa mukaan maailmanlaajuisiin järjestöihin, kuten esimerkiksi ISO ja IEC; kansainvälisiin järjestöihin, kuten esimerkiksi CEN ja CENELEC Euroopan alueella; sekä kansallisiin järjestöihin, kuten SFS Suomessa.

QAiRAn regulaatioseuranta tarjoaa siis asiakasyrityksilleen työkaluja ja dataa seurata kehittyvää lainsäädäntöä ja standardeja, sekä osoittaa vaatimustenmukaisuutensa osana tuotteistusprosessia. Kasven tuote-esittelysivujen sanoin QAiRA ”on tehty helpottamaan laatupäälliköiden, regulaatioista vastaavien henkilöiden ja tuotepäälliköiden työtä ja edesauttamaan tuotteen nopeampaa markkinoille viemistä” (Kasve).

4 KÄYTETYT TEKNIIKAT

Lähes kaikki opinnäytetyöprojektissa käytössä olleet teknologiat ovat olleet käytössä joko QAiRAn kehityksessä tai Severan tapauksessa Kasven muissa prosesseissa jo aiemmin. Ainoastaan Severan REST-rajapinta konfiguroitiin ja lisättiin vasta opinnäytetyöprojektin myötä uutena teknologiana QAiRAn kehitystyökalujen joukkoon.

Nykyisen QAiRAn todennäköisesti täysin ensimmäisistä versioista lähtien frontend-tason toiminnallisuudet on toteutettu JavaScriptillä ja jQueryllä, ja backend-taso PHP:llä käyttäen tietokantana joko MySQL:ää tai MariaDB:tä. Kehityksen jatkuessa prototyyppivaiheesta vuoden 2019 aikana projektissa otettiin käyttöön jaettu versionhallintatietovarasto GitLabin kautta, sekä projektinhallinta- ja dokumentointityökalut Jira ja Confluence. QAiRA julkaistiin asiakaskäyttöön saman vuoden keväällä. Myöhemmin käyttöön otettiin sähköpostipalvelu Mailgun sovelluksen automatisoitujen sähköpostiviestien lähetystä varten.

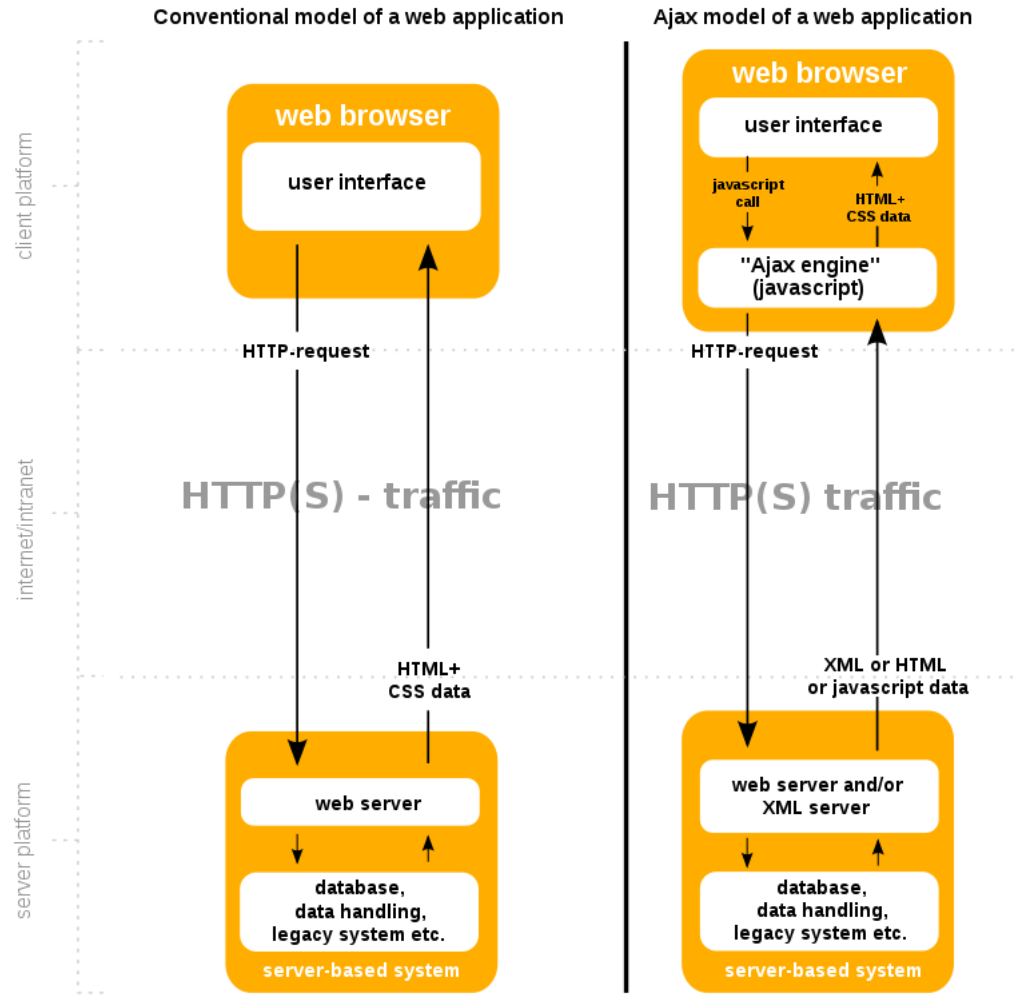
Severa on pitkään ollut käytössä Kasven muissa projekteissa muun muassa asiakkuuksien hallinnassa ja seurannassa. Opinnäytetyössä toteutetun asiakastietointegraation myötä sen on tarkoitus muodostua yhä keskeisemmäksi osaksi myös QAiRAn myyntiprosessien seurantaa.

4.1 PHP

PHP on erityisesti backend- eli palvelintason ohjelmointiin hyvin yleisesti käytetty ohjelmointikieli. PHP:n suosioon ovat vaikuttaneet muun muassa sen helppokäyttöisyys, tehokkuus, kattavuus eri käyttöjärjestelmien suhteen ja kyky tukea monille eri palvelimille rakennettuja ratkaisuja. (Carr & Gray 2018, 1.)

Verkkoteknologioiden käyttöä tutkivia mittauksia ja kartoituksia laativan W3Techs-sivuston mukaan PHP:tä käyttää palvelintason kielenään nykyisin noin 79% sivuston tarkkailemista verkkosivuista tehden siitä siis tällä hetkellä käytetyimmän palvelintason ohjelmointikielen (W3Techs).

PHP on tarkoitettu käytettäväksi erityisesti dynaamisesti latautuvien sivustojen rakentamiseen (PHP). Nykyisin erittäin yleinen menetelmä on hakea haluttu data Ajax-kutsulla palvelintasolta, jossa se prosessoidaan JSON-muotoon ja lähetetään frontend- eli pääteohjelmatasolle ja siellä prosessoidaan edelleen JavaScript-pohjaisilla teknologioilla käyttäjälle esitettävään muotoon (ks. kuva 3).



KUVA 3 Havainnollistus verkkosovelluksen Ajax-mallista (DanielSHaischt 2013, CC BY-SA 3.0)

Kielen ensimmäisten versioiden pohjina toimineet työkalut kehitti Rasmus Lerdorf vuosina 1994 ja 1995. Lerdorf julkaisi kehittämänsä työkalut avoimen lähdekoodin lisenssillä ja myöhemmin kehitystyöhön liittyi mukaan lisää tekijöitä. Projekti kehittyi hiljalleen enemmän kokonaisen ohjelmointikielen suuntaan ja mukaan liittyivät myös Andi Gutmans ja Zeev Suraski, jotka kehittivät kielen toiminnan kannalta tärkeän Zend-moottorin. Nykyään PHP:n kehityksestä ja ylläpidosta vastaa monikansallinen työryhmä, jossa Lerdorf, Gutmans ja Suraski ovat edelleen mukana. (PHP.)

4.2 JavaScript

JavaScript on alun perin verkkosivustojen frontend- eli pääteohjelmatason ohjelmointiin tarkoitettu ohjelmointikieli. JavaScript-koodi tulkitaan ja suoritetaan loppukäyttäjän verkkoselaimessa. Yhdessä HTML:n ja CSS:n kanssa JavaScript muodostaa kolminaisuuden, joka määrittää loppukäyttäjän vuorovaikutuksen verkkosivuston sisällön kanssa; HTML määrittää sisällön asettelun, CSS muotoilun ja JavaScript toiminnallisuuden.

JavaScript julkaistiin vuonna 1995 Netscape Navigator -selaimelle ja se on sittemmin kehitetty osaksi myös kaikkien nykyaikaisten verkkoselaimien toimintaa. Nykyään JavaScriptin kehitys on vahvasti

sidottu sitä määrittelevään ECMAScript-standardiin, joka on nimetty sen määritelleen standardointiyhdistyksen, Ecma Internationalin mukaan. JavaScriptin ja sen mukana kehittyvien selaimien uudistuksiin kiteytyy yksi, pohjimmiltaan kaiken verkkokehittämisen taustalla oleva päädilemma: teknologioita on kehitettävä vastaamaan uusiin mahdollisuuksiin ja käyttötarpeisiin, mutta kuitenkin niin, että olemassa olevat ohjelmat ja toteutukset kärsivät uudistuksista mahdollisimman vähän. (Haverbeke 2019, 5–6.)

Asia, joka JavaScriptiä esitellessä usein mainitaan, on, että se ei nimestään huolimatta ominaisuuksiltaan, toiminnallisuuksiltaan tai käyttötarkoituksiltaan liity juurikaan Javaan, täysin erilliseen ja eri tarkoitukseen kehitettyyn ohjelmointikieleen. JavaScriptin kehittämisen aikoihin suositun ja vahvasti markkinoidun Javan nimeä uskotaan yleisesti käytetyn lähinnä edistämään JavaScriptin suosiota. (Haverbeke 2019, 5.)

4.3 jQuery

jQuery (sic) on JavaScriptin kirjoittamisen helpottamiseksi ja nopeuttamiseksi kehitetty JavaScript-kirjasto. Toisin sanoen se on kokoelma erilaisia valmiiksi konfiguroituja luokkia ja metodeita, jotka lyhentävät tarvittavaa ohjelmakoodia, tehostavat sen suorittamista ja varmistavat sen standardinmukaisuuden. (York 2015, 3.)

Erityisesti jQueryn julkaisun aikaan yksi sen pääasiallisista tavoitteista on ollut myös taata koodin toiminnallisuus eri verkkoselaimissa. JavaScriptin tulkinnassa ja suorituksessa voi olla selainkohtaisia eroja ja selaimen kehittäjästä riippuu, miten JavaScriptin määrittävää standardia on seurattu ja tulkittu toteutuksessa. Etenkin nyt jo lakkautettu Internet Explorer on tunnetusti poikennut standardista hyvin paljon, mutta selaimen aiemman suosion vuoksi kehittäjien on pitkään pitänyt kuitenkin tukea sitä sovelluksissaan ja ottaa se huomioon kehitystyössä. (York 2015, 3.)

Nykypäivänä jQuery tarjoaa yhä monia hyödyllisiä toiminnallisuuksia, joiden ansiosta sen suosio on säilynyt. Tärkeimpinä jQueryn tarjoamina toiminnallisuuksina mainitaan usein mahdollisuus viitata HTML-sivun elementteihin niin kutsutuilla CSS-valitsimilla (CSS selectors), sekä useiden objektimetodien linkittäminen (ks. kuva 4). (York, 2015, 3–4.)

```
$('<div/>')
    .addClass('selected')
    .attr({
        id : 'body',
        title : 'Welcome to jQuery'
    })
    .text("Hello, World!");
```

KUVA 4 Esimerkki useiden objektimetodien linkittämisestä jQueryn avulla (York 2015, 4)

Vuodesta 2019 lähtien jQueryn kehitys on ollut osa OpenJS Foundationin toimintaa. OpenJS Foundation on säätiö, joka kehittää ja ylläpitää useita eri JavaScript kirjastoja. (OpenJS Foundation.)

4.4 MariaDB / MySQL

MariaDB on relaatiotietokantajärjestelmä (englanniksi RDBMS, Relational Database Management System), jonka tarkoituksena on tallentaa ja säilyttää dataa ja tietoa. MariaDB:n kaltaisessa relaatiotietokannassa data tallennetaan yksittäisiä objekteja tai kattokäsitteitä kuvaaviin kaksiulotteisiin tauluihin. Esimerkki tietokantataulusta voi olla vaikkapa *asiakas*. Taulun sarakkeet kuvaavat yksittäistä käyttäjästä tallennettavaa tietoa; *asiakkaan* tapauksessa yksittäisiä sarakkeita voivat olla esimerkiksi *etunimi* ja *sukunimi*. Jokainen taulun rivi taas kuvaa yksittäisen objektin, esimerkiksi yksittäisen *asiakkaan*, tietoja. Relaatiotietokannalle tyypillisesti MariaDB:tä voidaan hallinnoida ja sille tehdä kantakyselyjä SQL-ohjelmointikielellä. (Sriparasa 2014, 23–24, 29–30.)

Relaatiotietokantarakenne perustuu Edgar F. Coddin ehdottamaan relaatiomalliin. Usein relaatiotietokannan tauluihin tallennettava data on myös taulujen välillä keskenään linkittynyttä. Käytännössä eri taulujen rivien väliset viittaukset toteutetaan niinkutsutuilla viite- eli vierasavaimilla. Vierasavaimet ovat taulujen yksilöityjä kenttiä, joilla pystytään luotettavasti luomaan viiteyhteys yhden taulun yhdeltä riviltä toisen taulun riviin. (Sriparasa 2014, 23–24.)

MariaDB liittyy hyvin läheisesti toiseen vastaavaan relaatiotietokantajärjestelmään, MySQL:ään. MariaDB on MySQL:n kehitysryhmän – erityisesti sen pääkehittäjän, Michael ”Monty” Wideniuksen – kehittämä, uudempi tietokantajärjestelmä. MySQL:n oikeudet on sittemmin myyty Oracle Corporationille, tietokantoihin ja pilvipalveluihin erikoistuneelle tietotekniikkayhtiölle. MariaDB:n tarkoituksena taas on ollut tarjota MySQL:ää vastaava tai siitä edelleen kehitetympi tietokantajärjestelmä avoimen lähdekoodin lisenssillä. MariaDB:n kehityksestä vastaa voittoa tavoittelematon säätiö, MariaDB Foundation. Sekä MariaDB, että MySQL ovat nimetty Wideniuksen lasten, Marian ja Myn mukaan. (MariaDB.)

4.5 MailGun

Mailgun on sähköpostien lähetykseen ja seurantaan tarkoitettu palvelu, joka toimii REST-rajapinnan kautta. Käytännössä Mailgun toimii siis kehitettävässä sovelluksessa kolmannen osapuolen palveluna, jolla voidaan toteuttaa esimerkiksi automatisoitujen sähköpostiviestien tai uutiskirjeiden tyyppisten massapostitusten lähetystä ilman että kehitysryhmän tarvitsee asentaa omia sähköpostipalvelimia ja pitää huolta niiden lähetysmaineesta ja niin edelleen. (Mailgun.)

Mailgun perustettiin vuonna 2010 sen kehittäjien huomatessa, että vastaavia toiminnallisuuksia ei ollut saatavissa muista palveluista (Mailgun). Vuonna 2021 Mailgun oli yksi suositelluimmista SMTP-palveluntarjoajista.

4.6 Visma Severa

Visma on ohjelmistoyritys, joka kehittää kattavasti erilaisia digitaalisia palveluita eri markkinasektorien yritysten ja organisaatioiden käyttöön. Visman omien kotisivujen mukaan ” Visma kehittää ja myy ohjelmistoja, jotka yksinkertaistavat ja digitalisoivat yritysten ja julkishallinnon organisaatioiden keskeisiä liiketoimintaprosesseja. – – Visma-konserni koostuu yli 200 yrityksestä yli 20 maassa ympäri maailmaa.” (Visma.)

Severa on Visman kehittämä projektinhallintajärjestelmä, joka tarjoaa työkaluja muun muassa asiakkuuksien hallintaan, myynnin sekä työajan seurantaan, resursointiin, laskutukseen ja raportointiin (Visma). Asiakkuuksien hallinnan osalta järjestelmään on mahdollista kirjata asiakasorganisaatioita, niiden tietoja sekä yhteyshenkilöitä ja heidän roolejaan, sekä tietojaan. Asiakasyrityksiin voidaan liittää erilaisia asiakasryhmätietoja ja yhteyshenkilöihin avainsanoja, joiden avulla voidaan muun muassa kartoittaa asiakkaalle myytyjä palveluita ja ryhmitellä yhteyshenkilöitä esimerkiksi markkinointiviestien postitusta varten. Visma Severa tarjoaa myös REST-rajapinnan, joka mahdollistaa automatisoitujen tiedonhaku- ja päivitystoimintojen rakentamisen Severan ja sen ulkopuolisisten sovellusten välille (Visma).

4.7 Jira ja Confluence

Atlassian on vuonna 2002 perustettu ohjelmistoyritys, joka tuottaa ketterään kehitykseen tarkoitettuja projektinhallintaohjelmia ja -järjestelmiä. Jira ja Confluence ovat Atlassianin pitkäaikaisimpia ja kenties tunnetuimpia tuotteita. (Atlassian.)

Jira on tehtävänhallintasovellus, joka soveltuu erityisesti ketterän kehityksen Scrum- ja Kanban-malleille (Harned 2018, 5). Yleensä ketterän kehityksen malleissa kehitysaikataulu jaetaan esimerkiksi kahden viikon pituisiin sprintteihin, joihin voidaan ottaa erilaisia kehitys-, testaus-, dokumentointi- ja virheiden korjaustehtäviä. Jirassa voidaan laatia, seurata ja raportoida sprintteihin perustuvia kehi-

tyssuunnitelmia monella tasolla pidemmän tähtäimen liiketoimintasuunnitelmista yksittäisiin kehitystehtäviin ja lopullisiin versiojulkaisuihin. Sovelluksessa on myös mahdollista roolittaa tehtävät yksittäisille kehittäjille ja laatia erilaisia julkaisuaikatauluja sekä yksilöityjä työnkulkukaavioita (Atlassian).

Confluence on wikipohjainen dokumentinhallintasovellus, jolla voidaan luoda, muokata, jakaa ja säilöä erilaisia kehitystyössä tarvittavia ja syntyviä dokumentteja, muistioita ja tehtävälistoja. Confluence tarjoaa useita valmiiksi mallinnettuja dokumenttipohjia ja tekstinkäsittelytyökaluja dokumenttien laatimiseen. Jira ja Confluence ovat myös keskenään yhteydessä, niin että Confluence-dokumenteissa voidaan tehdä viittauksia Jira-tehtäviin ja Jira-tehtävissä vastaavasti Confluence-dokumentteihin. (Atlassian.)

4.8 Git

Git on vapaa, avoimen lähdekoodin, hajautettu versionhallintajärjestelmä (Git). Gitin avulla kehittäjän on mahdollista tallentaa sovelluksen tiedostomuutoksia tiettyinä ajanhetkinä, siirtää muutoksia esimerkiksi paikallisesta kehitysympäristöstä jaettuun ympäristöön ja tarvittaessa palata myös takaisin aikaisempaan versioon (Geisshirt, Olsson, Voss, & Zattin 2018, 1).

Gitin hajautetun rakenteen ansiosta useampi kehittäjä voi työskennellä samojen tiedostojen kanssa samanaikaisesti. Käytännössä kehittäjä voi ”kloonata” nykyisen version jaetusta tietovarastosta (eng. repository) paikalliseen ympäristöönsä ja luoda siitä uuden kehitysoksan (eng. branch). Hän voi sen jälkeen jatkaa kehitystä omassa oksassaan, kunnes toiminnallisuus on valmis, jolloin oksan tiedostoihin tehdyt muutokset voidaan yhdistää jaetun tietovaraston pääoksaan.

Mikäli tiedostoihin on tehty samojen versioiden välillä useampia eri muutoksia samaan kohtaan, luovat ne konflikteja, jotka on ratkaistava ennen kuin tiedostoversioiden yhdistys voidaan saattaa loppuun. Käytännössä kehittäjä tai asianosainen versioyhdistyspyynnön tarkastaja käy konfliktit läpi yksi kerrallaan ja valitsee, mitkä muutokset eri tiedostoversioista säilytetään.

Gitin on alun perin kehittänyt Linus Torvalds vuonna 2005 (Git). Git kehitettiin korvaamaan Linux-käyttöjärjestelmän kehityksessä käytetty yksityisomistuksellinen versionhallintajärjestelmä. Sittemmin Gitistä on tullut vakioratkaisu useimpien nykyaikaisten sovelluskehitysprojektien versionhallintaan. (Geisshirt ym. 2018, 1.)

4.9 GitLab

GitLab on Git-tietovarastojen hallintaan kehitetty järjestelmä. Se tarjoaa käyttäjilleen jaettujen tietovarastojen palvelinylläpidon, sekä graafisen käyttöliittymän muun muassa tietovarastojen, oksien ja käyttäjien hallintaan, dokumentointiin sekä versiomuutosten yhdistykseen ja seurantaan. (Hethey 2013, 5–7.)

GitLab on avoimen lähdekoodin projekti, joka on saanut alkunsa vuonna 2011. Sillä on arviolta yli 30 miljoonaa rekisteröityä käyttäjää ja sen kehitysyhteisöön kuuluu yli 2500 avustavaa kehittäjää. GitLab-yhtiö on niinkutsuttu open-core-mallin yhtiö, jonka tuote perustuu avoimeen lähdekoodiin, mutta joka myy maksaville käyttäjilleen lisäominaisuuksia. (GitLab.)

Aiemmin GitLabin selkeänä etuna kilpailijoihinsa nähden oli, että se tarjosi yksityisesti jaettavia tietovarastoja myös ilmaisessa tilaussuunnitelmassaan (Hethey 2013, 8). Myöhemmin kuitenkin myös sen kanssa kilpaileva, suosittu GitHub-palvelu on vastaavasti alkanut tarjoamaan yksityisiä tietovarastoja omassa ilmaisessa suunnitelmassaan (GitHub). GitHub tosin on yksityisomistuksellinen eli suljetun lähdekoodin ohjelmisto, mutta on olemassa myös muita vastaavia avoimen lähdekoodin versionhallintajärjestelmiä (Hethey 2013, 8).

5 TOTEUTUS

Opinnäytetyö koostuu kolmesta opinnäytetyöperiodin aikana QAiRAan uuskehityksenä toteutetusta toiminnallisuudesta, niiden testauksesta ja dokumentoinnista. Toiminnallisuudet on toteutettu käyttäen QAiRA-projektissa yleisesti käytettyjä frontend- ja backend-teknologioita. Kehitysprosessi ja työnkulku on dokumentoitu Jiraan ja toiminnallisuudet itsessään on dokumentoitu Confluenceen. Kolme mainittua toiminnallisuutta ovat

1. Regulaatioaineistopäivitysdatan lisäys osaksi automatisoitua sisältöpäivityskoostetta
2. Automatisoitujen sähköpostien lähetystyökalu
3. Asiakaskäyttäjätietointegraatio.

Toiminnallisuudet palvelevat keskenään hyvin erilaisia käyttäjäryhmiä sekä QAiRAn että Kasven sisällä. Ensimmäinen toiminnallisuus on tarkoitettu QAiRAn asiakasorganisaatioiden eli niin kutsuttujen loppukäyttäjien käyttöön, toinen on tehty QAiRA-järjestelmäylläpitäjille eli sovelluksen sisäiseen ylläpitoon ja virhetilanteiden ratkontaan ja kolmas Kasven sisäiseen QAiRAn markkinointi- ja liiketoimintadatan seurantaan ja hallinnointiin.

5.1 Regulaatioaineistopäivitysdatan lisäys osaksi automatisoitua sisältöpäivityskoostetta

Ensimmäinen opinnäytetyön tehtävistä oli niin sanotun regulaatioaineistopäivitysdatan lisääminen osaksi automatisoitua sisältöpäivityskoostetta eli niin kutsuttua QAiRA Newsletteriä. Käyttäjän määrittämistä asetuksista riippuen QAiRA-sovellus siis lähettää käyttäjälle viikottain sähköpostitse koosteen viikon aikana ilmestyneistä uutisartikkeleista. Sisältöpäivityskooste pitää sisällään käyttäjän seuraamien uutislähteiden otsikoita, linkkejä ja tiivistelmiä.

Käytännössä järjestelmä suorittaa säännöllisin väliajoin skriptin, joka vertailee uutislähteistä haettua ja tietokannassa jo olemassa olevaa uutisdataa keskenään ja tekee tarvittavat lisäykset kantaan. Uutisia voi lisäyksen jälkeen selata QAiRA-sovelluksessa ja käyttäjä voi valita uutisasetuksista myös haluavansa vastaanottaa tiedotteita eli notifikaatioita eri uutislähteistä (ks. kuva 5). Kerran viikossa järjestelmä suorittaa toisen skriptin, joka koostaa käyttäjän valintojen mukaan jokaiselle käyttäjälle yksilöidyn koosteen viikon aikana ilmestyneistä uutisista.

Tehtävänä oli siis laatia samanlainen toiminnallisuus regulaatioiden seurantaan, jotta käyttäjät voisivat vastaanottaa sisältöpäivityskoosteessa tiedotteita päivittyneistä standardeista. Toteutettu toiminnallisuus on kuvattu seuraavissa kappaleissa. Toiminnallisuuden toteuttaminen vaati seuraavat kolme vaihetta:

1. Regulaatioaineiston päivityshistorian tallennus tietokantaan
2. Käyttöliittymäkomponenttien lisäys, jotta käyttäjä voi valita vastaanottavansa tiedotteita seuraamistaan standardeista
3. Regulaatioiden päivityshistorian koostaminen osaksi sisältöpäivityskoostetta.

KUVA 5 Käyttäjän uutisasetukset, josta käyttäjä voi valita vastaanottavansa tiedotteita QAiRAn uutislähteistä (uutislähteiden tiedot on häivytetty kuvasta)

5.1.1 Regulaatioaineiston päivityshistoria tietokannassa

Regulaatioaineiston päivityshistoria lisätään kannassa päivityshistoriatauluun (ks. kuva 6). Taulun yksi rivi kuvaa yhtä muutosta yhdessä regulaatiossa. Riviin tallennetaan regulaation uniikki id-numero toisessa taulussa, regulaation tyyppi (lainsäädäntö, ohjeistus tai standardi), tarvittaessa statustiedot tehdyistä muutoksista (toisin sanoen mikä oli regulaation tila ennen muutoksia ja sen jälkeen), sekä tehtyjen muutoksien päivämäärä. Regulaation id-numeron ja tyypin avulla saadaan tarvittaessa koottua kaikki yksittäisen regulaation historiatiedot. On syytä huomata, ettei yhdistelmä ole taulussa uniikki, sillä ajan myötä yhteen regulaatioon saattaa tulla useampia muutoksia.

	#	Name	Type	Collation	Attributes
☐	1	id	int(11)		
☐	2	regulationId	int(11)		
☐	3	regulationType	tinyint(4)		
☐	4	statusOld	varchar(255)	utf8mb4_swedish_ci	
☐	5	statusNew	varchar(255)	utf8mb4_swedish_ci	
☐	6	changeDate	datetime		

KUVA 6 Päivityshistoriataulun sarakkeet tietokannassa

Tauluun voi lisätä tietoja manuaalisesti, mutta automatisoitujen päivitysprosessien osalta myös historiatietojen tallennus kannattaa lisätä osaksi automaatiota. QAiRAssa olevista regulaatioista standardiaineiston, sekä niin kutsuttujen FDA-ohjeistuksien päivitys on osittain automatisoitu. Kummankin aineiston päivitysprosessi on samantyyppinen. Päivityksessä käytetään kahta skriptiä.

Ensimmäinen skripti hakee nykyisen aineiston lähteestä, vertaa sitä tietokannassa olevaan aineistoon ja koostaa aineistojen erojen perusteella tarvittavat muutokset regulaatiotyypistä riippuen yhteen tai useampaan CSV-tiedostoon. Tiedoston riveille tallennetaan regulaatiotyypistä riippuen tarvittavat tiedot, kuten esimerkiksi regulaation nimi, koodi, sisältö, tiivistelmä ja niin edelleen. Muiden tietojen lisäksi riville tallennetaan myös tieto siitä, halutaanko regulaatio päivittää, lisätä tietokantaan vai poistaa sieltä.

On kolme eri tilannetta, jotka tuottavat rivejä tiedostoon. Jos regulaatio löytyy lähdeaineistosta, mutta ei löydy tietokannasta, rivi on lisättävä tietokantaan. Jos regulaatio löytyy vain tietokannasta, mutta ei lähdeaineistosta, kirjataan regulaatio poistettavaksi tietokannasta. Jos taas regulaatio löytyy sekä lähdeaineistosta että tietokannasta, mutta sen tiedot poikkeavat eri versioissa toisistaan, on lähdeaineistosta saadut tiedot päivitettävä tietokantaan. Päivityshistoriatietojen kannalta viimeksi mainitut muutokset ovat olennaisia; mikäli jonkin regulaation tiedot päivitetään, on päivityksestä lisättävä myös rivi päivityshistoriatauluun kannassa.

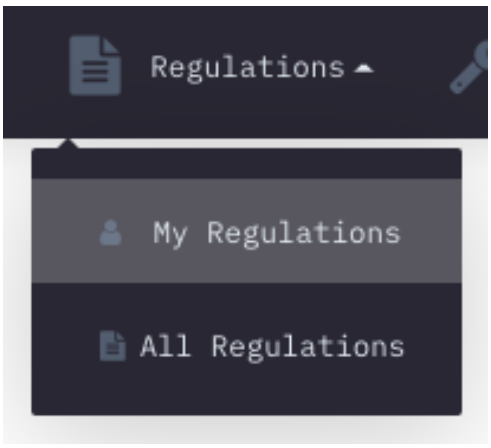
Ensimmäisen skriptin muodostettua CSV-tiedoston voidaan sisältöä tarkistaa manuaalisesti pistokokein tai tarkemmin läpikäymällä esimerkiksi Excelin tai vastaavan taulukko-ohjelman avulla. Kun sisältö halutaan päivittää tietokantaan, suoritetaan palvelimella toinen skripti, joka ottaa luodun CSV-tiedoston syötteenä ja käy sen läpi rivi kerrallaan tehden tietokantaan tarvittavat muutokset mukaan lukien rivien lisäykset muutoshistoriatauluun. Muutoshistoriatiedot on koodissa mallinnettu luokalla, jonka ominaisuudet vastaavat CSV-tiedoston sarakkeita (ks. kuva 7). Luokalla on myös metodi, jolla sen tiedot voidaan tarvittaessa syöttää taulukkorakenteeseen.

```php
<?php
class RegulationHistoryLog {
    // Properties
    public $id;
    public $regulation_id;
    public $regulation_type;
    public $status_old;
    public $status_new;
    public $change_date;
    public $action;

    // Methods
    function __construct($props) {
        // --- Mandatory fields ---
        $this->regulation_id    = $props['regulation_id'];
        $this->regulation_type  = $props['regulation_type'];
        $this->status_old       = $props['status_old'];
        $this->status_new       = $props['status_new'];

        // --- Optional fields ---
        $this->id       = isset($props['id']) ? $props['id'] : '';
        $this->action   = isset($props['action']) ? $props['action'] : '';
    }

    // Function to turn the object into an associative array
    function toArray() {
        // Initialize the return array
          $arr = [];
          // Loop through this Standard object and fill the array
          foreach ($this as $k => $v) {
              $arr[$k] = $v;
          }
          return $arr;
    }
```

KUVA 7 Regulaation päivityshistoriaa mallintava luokka

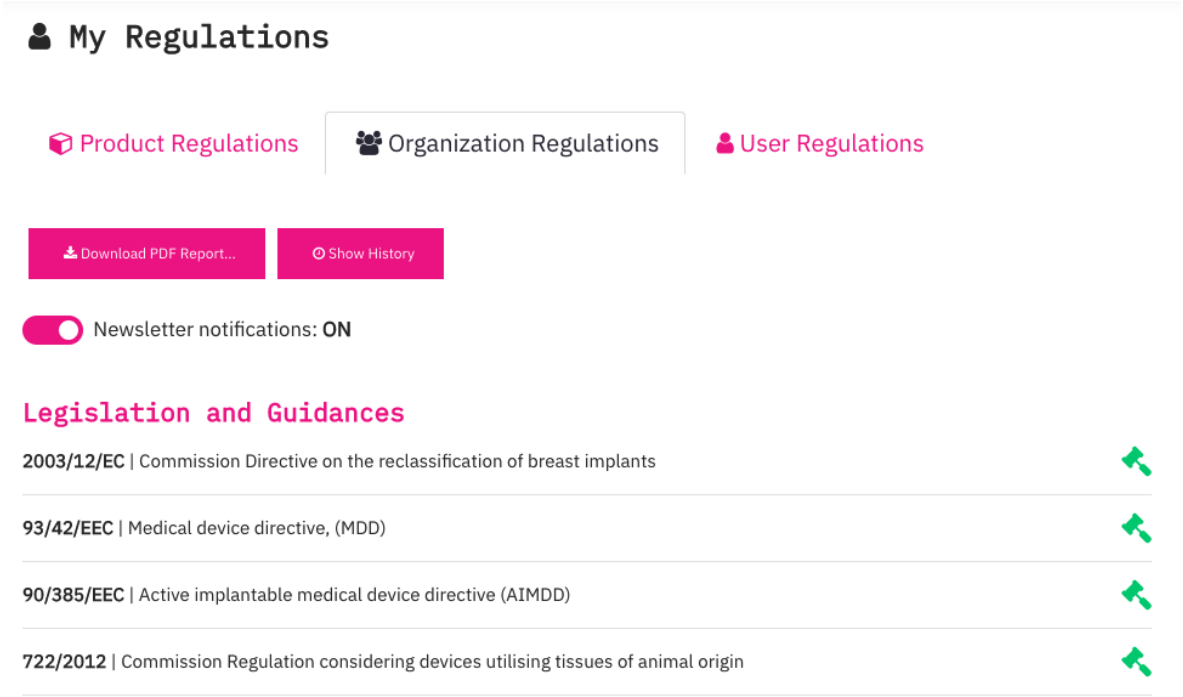
5.1.2 Tiedoteasetusten käyttöliittymäkomponentit

Regulaatioiden seurantaan liittyvät toiminnallisuudet on QAiRAssa jaettu kahdelle sivulle: All Regulations ja My Regulations (ks. kuva 8). All Regulations sisältää regulaatioiden hakuun, suodatukseen ja yleiseen tutkimiseen tarkoitettuja työkaluja. All Regulations -sivun kautta käyttäjä voi lisätä tarvitsemiaan regulaatioita My Regulations -sivulle.



KUVA 8 Regulaatioiden seurantasivut QAiRAn navigaatiovalikossa

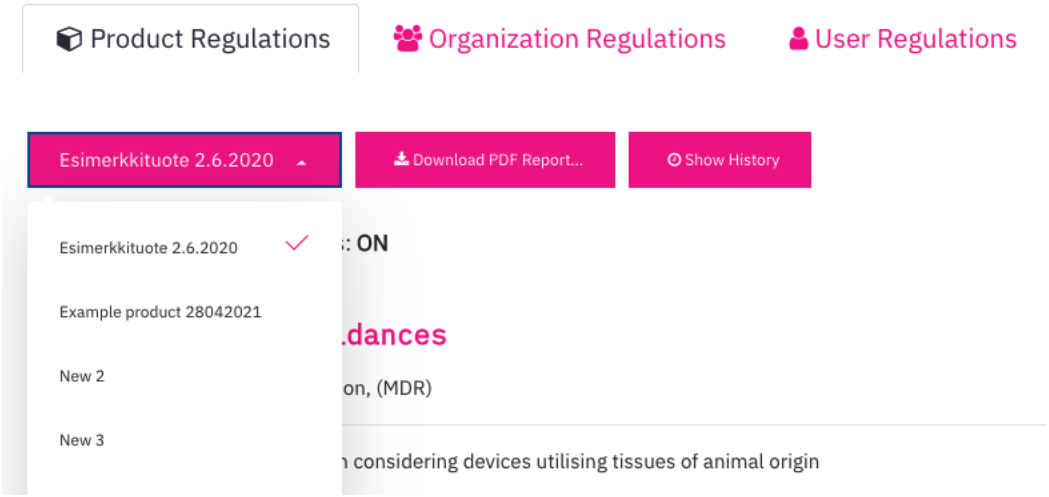
Kahdesta edellä mainitusta sivusta My Regulations on sisältöpäivityskoosteen kannalta olennainen sivu. My Regulations sisältää listoja käyttäjien seuraamista regulaatioista, joihin organisaation käyttäjät voivat itse valita tarvitsemansa regulaatiot (ks. kuva 9). QAiRAssa on mahdollista seurata ja ryhmitellä regulaatioita kolmella eri tasolla, jotka on eroteltu My Regulations -sivun välilehtiin. Mainitut tasot ovat tuote-, organisaatio- ja käyttäjäkohtaiset regulaatiot.



KUVA 9 Esimerkki My Regulations -sivun näkymästä organisaatiovälilehdellä

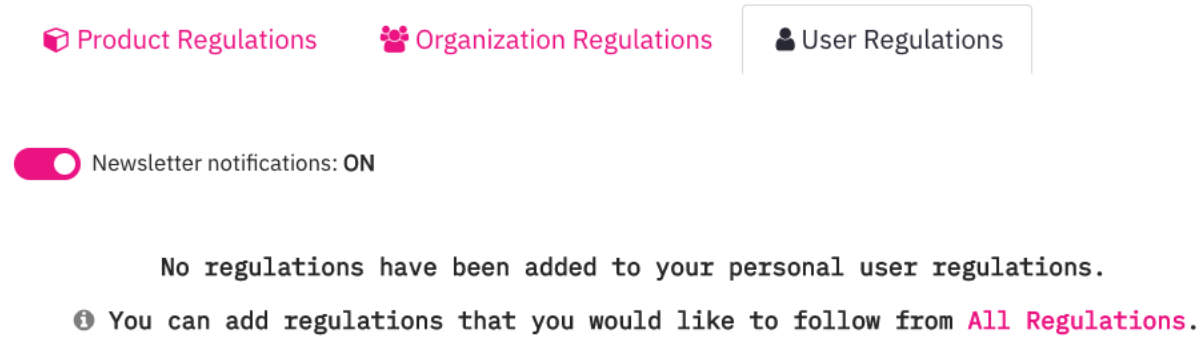
My Regulations -sivun oletusnäkymä aukeaa organisaatiokohtaisista regulaatioista, jotka ovat yhteisiä organisaation kaikille käyttäjille. Kaikki organisaation käyttäjät näkevät siis samat regulaatiot organisaatiovälilehdellä ja kaikki organisaation pääkäyttäjät voivat lisätä ja poistaa regulaatioita organisaation listasta.

Tuotevälilehti sisältää alasvetovalikon, joka listaa kaikki organisaation tuotteet, joihin käyttäjällä on katseluoikeudet (ks. kuva 10). Jokaisella tuotteella on oma regulaatiolistansa, jota voivat muokata kaikki käyttäjät, joilla on muokkausoikeudet tuotteeseen.



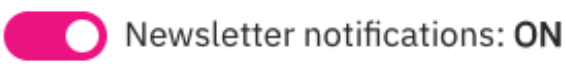
KUVA 10 Esimerkki My Regulations -sivun näkymästä tuotevälilehdellä

Käyttäjävälilehti sisältää listan käyttäjän henkilökohtaisesti seuraamista regulaatioista (ks. kuva 11). Muut käyttäjät eivät näe toistensa henkilökohtaisia listoja. Vain käyttäjä itse voi muokata omaa regulaatiolistaansa.

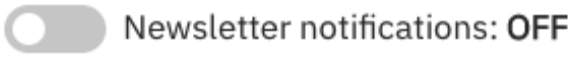


KUVA 11 Esimerkki My Regulations -sivun näkymästä käyttäjävälilehdellä

Sisältöpäivityskoosteen uudistuksen myötä lisätty komponentti on jokaisen välilehden alussa oleva ”sisältöpäivityskoosteen tiedotteet” -kytkin (Newsletter notifications). Käyttäjä voi asettaa kytkimen joko ON- tai OFF-asentoon erikseen henkilökohtaisen regulaatiolistansa, organisaatioregulaatioiden, sekä jokaisen näkemänsä tuotteen kannalta (ks. kuvat 12 ja 13). Kytkimen tila päivittyy tilanteen mukaan välilehteä tai tuotetta vaihtaessa.



KUVA 12 Sisältöpäivityskoosteen tiedotteet -kytkin päällekytkettynä



KUVA 13 Sisältöpäivityskoosteen tiedotteet -kytkin kytkettynä pois päältä

Kytkin kuvaa asetusta siitä, lisätäänkö viikottain lähetettävään sisältöpäivityskoosteeseen mahdolliset päivitystiedot käyttäjän henkilökohtaisesti seuraamista tai käyttäjän organisaation tai seuraamien tuotteiden regulaatioista. Kytkimen tilan muutos tekee AJAX-prosessin kautta POST-kutsun palvelintasolle. Kutsun tiedot lähetetään JSON-muodossa käyttäen pohjana objektia, joka sisältää kytkimen tilan, regulaatiolistan tyypin (tuote, organisaatio vai käyttäjä), sekä tarvittaessa tuotteen yksilöivän id-arvon (ks. kuva 14).

```
data: {
    'value'          : value,
    'type'           : type,
    'product_id'     : product_id
},
```

KUVA 14 Kytkimen kautta lähetettävän POST-kutsun tiedot sisältävä objekti

Palvelintasolla vastaanotettu data prosessoidaan ja tehdään sen perusteella tarvittava muutos tietokantaan. Käyttäjän henkilökohtaista regulaatiolistaa koskeva tiedoteasetus päivitetään käyttäjätietoja sisältävään tauluun. Tuotteita koskevat asetukset päivitetään tuotteet ja käyttäjät toisiinsa linkittävään tauluun; näin eri käyttäjille on mahdollista asettaa eri asetuksia samalle tuotteelle. Käyttäjien ja tuotteiden väliset yhteydet on kuvattu niiden välisellä viitetaululla, koska kyseessä on niin kutsuttu monesta-moneen-yhteys, toisin sanoen yhdellä käyttäjällä voi olla oikeuksia moneen eri tuotteeseen ja yhtä tuotetta voi hallinnoida monta eri käyttäjää. Myös organisaatiota koskevat asetukset tallennetaan edellä mainittuun tuotteiden ja käyttäjien väliseen viitetauluun, koska organisaation regulaatiolista on kannassa mallinnettu organisaatioon sidotulla oletustuotteella.

5.1.3 Sisältöpäivityskoosteen muodostaminen ja lähetys

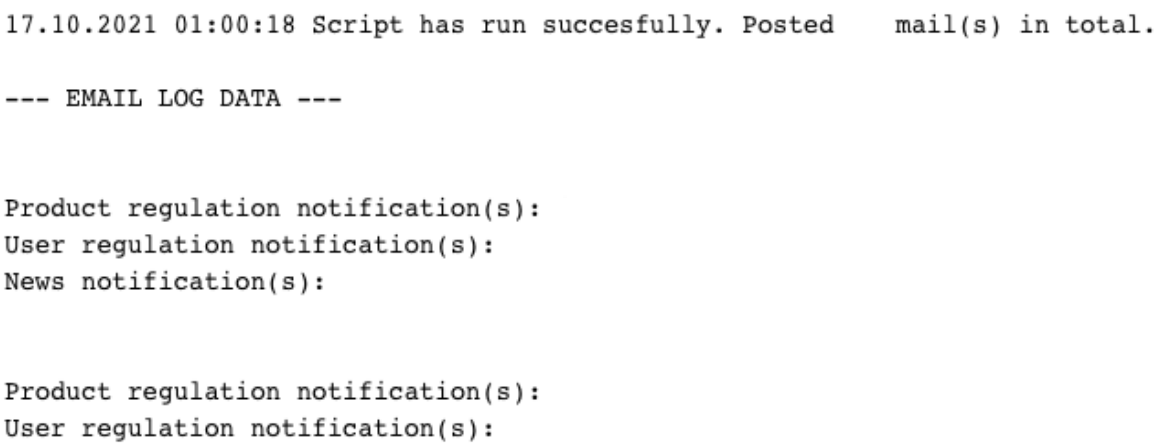
Sisältöpäivityskoosteen viikottain muodostava ja lähettävä skripti on myös uudistettu osana toiminnallisuuden toteutusta. Palvelinympäristö on konfiguroitu suorittamaan skripti tiettynä aikana kerran viikossa.

Suoritettaessa skripti hakee tietokannasta kaikkien aktiivisten käyttäjien tiedot. Skripti iteroi käyttäjälistan läpi käyttäen uutiskirjeen lähetykseen laadittua funktiota, joka käyttäjien yksilöityjen tunnisteiden perusteella koostaa ja lähettää kullekin käyttäjälle yksilöidyn sisältökoosteen. Mikäli käyttäjä

on esimerkiksi kytkenyt tiedoteasetuksen päälle tuote-, organisaatio- tai henkilökohtaisen regulaatiolistauksensa osalta, skripti hakee tietokannasta kyseisten listausten regulaatioiden päivitystiedot viikon ajalta ja lisää ne osaksi lähetettävän koosteen sisältöä (ks. kuva 15).

KUVA 15 Esimerkki organisaatiokohtaisten regulaatioiden standardilistauksesta sisältöpäivityskoosteessa. Päivittyneet standardit on listattu linkkeineen viestin sisältöön.

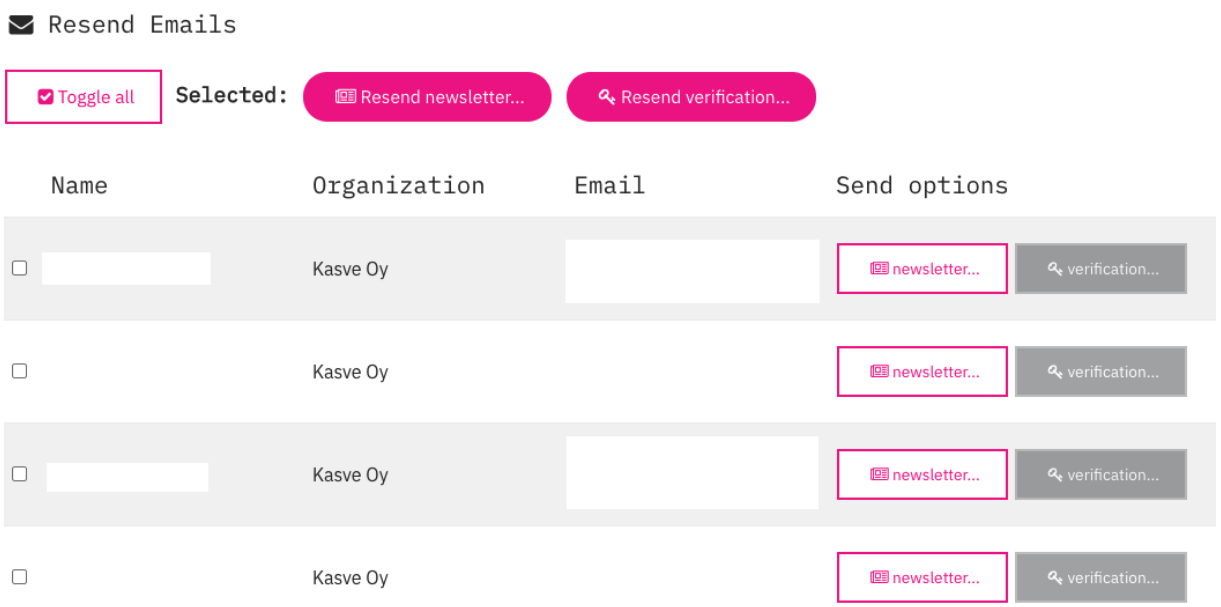
Sisältöpäivityskoosteen koostamis- ja lähetysfunktioiden teknisistä yksityiskohdista kerrotaan hieman tarkemmin luvussa *5.2.3. Sähköpostien lähetyksen backend-toteutus*. Koostettuaan viestin skripti lähettää sen käyttäjälle ja siirtyy prosessoimaan seuraavan käyttäjän tietoja. Skripti tallettaa suorituksensa ajan myös lokitietoja onnistuneista lähetyksistä ja virhetilanteista ja iteroituaan kaikki käyttäjät läpi kirjoittaa ne erilliseen lokitiedostoon (ks. kuva 16).



KUVA 16 Esimerkki sisältöpäivityskoosteiden lähetyksistä muodostetun lokitiedoston sisällöstä (yksilöivät tiedot on häivytetty kuvasta)

5.2 Automatisoitujen sähköpostien lähetystyökalu

Automatisoitujen sähköpostien lähetystyökalu on QAiRA-järjestelmäylläpitäjälle suunniteltu työkalu, jonka avulla ylläpitäjä voi lähettää QAiRA-järjestelmän luomia sähköpostiviestejä (ks. kuva 17). Laaditussa toteutuksessa on mahdollista lähettää QAiRAn sisältöpäivityskoosteviestejä eli niin kutsuttuja QAiRA Newslettereitä, sekä alun perin käyttäjän rekisteröityessä lähetettäviä sähköpostiosoitteen vahvistusviestejä.



KUVA 17 Automatisoitujen sähköpostiviestien lähetystyökalun käyttöliittymä (käyttäjien tiedot on häivytetty kuvasta)

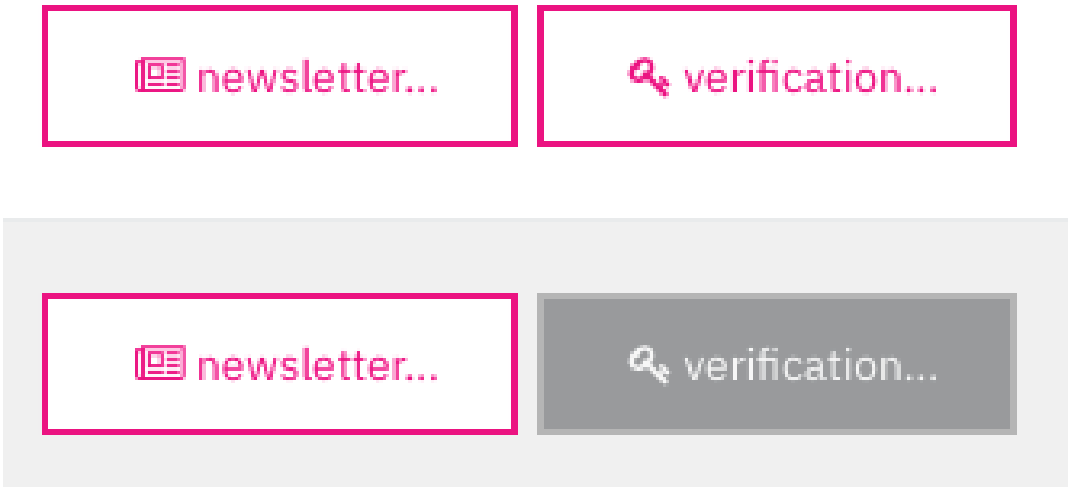
Kasve vastaa QAiRAn järjestelmäylläpidosta, joten työkalu on tarkoitettu Kasven ja QAiRAn kehitysryhmän sisäiseen käyttöön. Työkalun kaksi pääasiallista käyttötarkoitusta ovat viestitoiminnallisuuksien testaus ja virhetilanteiden ratkaisu tuotantoympäristössä.

5.2.1 Lähetystyökalun käyttöliittymä

Päästäkseen käyttämään työkalua käyttäjällä pitää olla QAiRA-järjestelmäylläpitäjän oikeudet. Sivusto käyttää yleisesti kehitysprojektissa jaettua funktiota tarkistaakseen käyttäjän oikeudet. Kun käyttäjäoikeudet on varmennettu, työkalun käyttöliittymän sivusto latautuu käyttäjän käytettäväksi.

Työkalu hakee kannasta kaikki QAiRAn käyttäjät ja listaa ne organisaation mukaan ryhmiteltynä aakkosjärjestyksessä. Työkalu listaa käyttäjän nimen, organisaation ja sähköpostin, joiden avulla käyttäjä voi hakea etsimänsä käyttäjän listasta. Työkalulla on mahdollista lähettää viestejä joko yksittäiselle käyttäjälle tai useammalle käyttäjälle kerrallaan.

Halutessaan lähettää viestin vain yksittäiselle käyttäjälle, pääkäyttäjä voi käyttää listan riveillä olevia painikkeita. Jokaisen käyttäjän rivillä on kaksi painiketta, nimellisesti ”newsletter” ja ”verification” (ks. kuva 18).



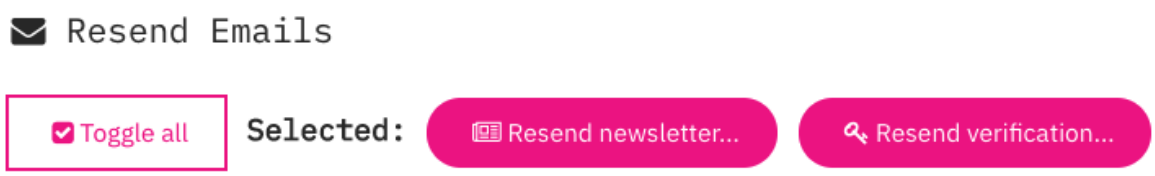
KUVA 18 Viestin lähetyspainikkeet käyttäjälistassa. Verification- eli sähköpostiosoitteen vahvistusviestin lähetyspainike voi olla kytketty päälle tai pois päältä riippuen käyttäjän tiedoista.

Newsletter-painike on aiemmin kuvaillun QAiRAn sisältöpäivityskoosteviestin lähettämistä varten. Sen pääasiallisena tarkoituksena on sisältöpäivityskoosteeseen liittyvien ja kehitettävien toiminnallisuuksien testaaminen. Sisältöpäivityskoosteet lähetetään tavallisesti vain kerran viikossa automaattisesti ja keskitetysti samaan aikaan kaikille QAiRAn käyttäjille, mutta kehittäessään uutta ominaisuutta tai ratkoessaan virhetilannetta kehittäjä voi haluta testata sisältöpäivityskoosteen muodostusta ja lähetystä lähettäen sen vain itselleen tai muulle yksittäiselle käyttäjälle.

Verification-painikkeen tarkoituksena on lähettää käyttäjälle sähköpostiosoitteen vahvistusviesti. Rekisteröityessään QAiRAan käyttäjä tai käyttäjän luonut organisaatiopääkäyttäjä syöttää järjestelmään muiden tietojen ohessa käyttäjän sähköpostiosoitteen. Järjestelmä sen jälkeen lähettää käyttäjälle automaattisesti vahvistusviestin, jonka avulla käyttäjä voi todistaa olevansa sähköpostiosoitteen todellinen omistaja. Sähköpostiosoitteen vahvistaminen on sovelluksen tietoturvallisuutta parantava ominaisuus. Toisaalta ilman sähköpostiosoitteen vahvistamista käyttäjä ei pääse kirjautumaan sisälle sovellukseen, joten vahvistusviestin saavutettavuus on myös ensiarvoisen tärkeää. Vahvistusviestin uudelleenlähetystoiminnallisuuden pääasiallinen tarkoitus on siis ratkaista virhetilanne, jossa käyttäjä ole vastaanottanut vahvistusviestiä tai on mahdollisesti hävittänyt alkuperäisen viestin tai muuta vastaavaa. Vastaanottajalistassa painike on kytketty käytettäväksi tai pois käytöstä riippuen siitä, onko käyttäjä jo vahvistanut sähköpostiosoitteensa vai ei (ks. kuva 18).

Halutessaan lähettää viestejä useammalle käyttäjälle samaan aikaan, pääkäyttäjä voi valita vastaanottajat listasta nimien edessä olevilla valintalaatikoilla ja sen jälkeen käyttää listan yläpuolelta löytyviä lähetyspainikkeita, jotka koostavat ja lähettävät viestit kaikille valituille käyttäjille (ks. kuva 19). ”Resend newsletter…” -painike lähettää sisältöpäivityskoosteen kaikille valituille vastaanottajille ja ”Resend verification…” -painike vastaavasti lähettää sähköpostiosoitteen vahvistusviestin kaikille va-

lituille käyttäjille, jotka sen voivat vastaanottaa. ”Toggle all” -painike muuttaa kaikkien valintalaatikoiden valinnat päinvastaisiksi, joten sitä voi käyttää valitakseen tai suodattaakseen suuren määrän käyttäjiä kerralla.

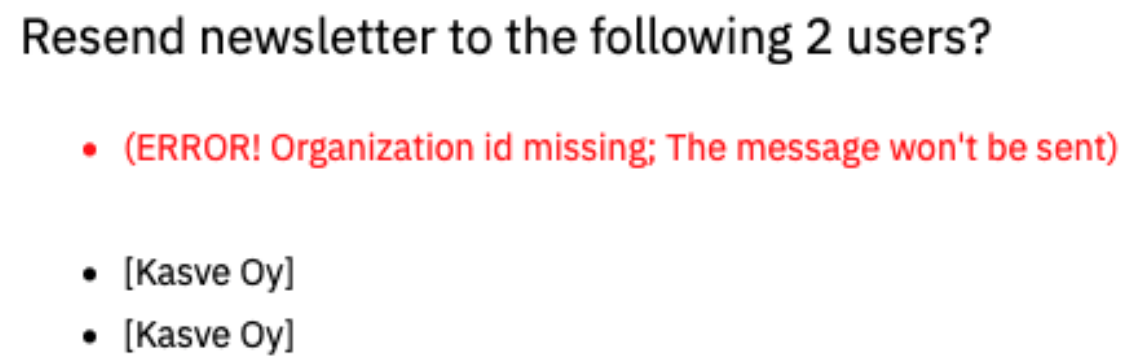


KUVA 19 Listan yläpuolella olevat valintalähetyspainikkeet

5.2.2 Lähetysmodaali

Kaikki mainitut lähetyspainikkeet avaavat modaalin, jossa on kerrattuna lista vastaanottajien organisaatiosta ja sähköpostiosoitteista (ks. kuva 20). Virhetilanteissa lista ilmoittaa, että viestejä ei lähetetä käyttäjille, joiden tiedot ovat jostain syystä puutteelliset (ks. kuva 21).

KUVA 20 Sisältöpäivityskoosteen lähetysmodaali (vastaanottajien sähköpostiosoitteet on häivytetty kuvasta). Sähköpostiosoitteen vahvistusviestin lähetysmodaali on samanlainen, mutta ilman ”Set the dates (optional)” -painiketta.



KUVA 21 Vastaanottajalistassa on virheilmoitus niiden käyttäjien kohdalla, joille ei voida lähettää viestejä puutteellisten tietojen takia (vastaanottajien sähköpostiosoitteet on häivytetty kuvasta).

Sisältöpäivityskoosteen (newsletter) lähetysmodaali sisältää myös kaksi painikkeesta avautuvaa vapaaehtoisesti täytettävää kenttää, joihin pääkäyttäjä voi syöttää päivämäärät, joiden väliltä haluaa koostaa uutiskirjeen tiedot (ks. kuva 22). Koosteen alkupäivämäärän oletusarvo on viikko nykyisestä päivästä ja lopetuspäivämäärän oletusarvo on tämä päivä, mutta testauksessa saatetaan haluta käyttää oletuksesta eroavia aikavälejä. Mikäli pääkäyttäjä jättää kentät tyhjiksi, käytetään koosteessa kenttien oletusarvoja. Tarkastettuaan viestien vastaanottajat pääkäyttäjä voi päättää lähettää viestit tai peruttaa lähetyksen modaalin alakulmista löytyvistä ”Save”- ja ”Cancel” -painikkeista.

KUVA 22 Sisältöpäivityskoosteen lähetysmodaali (vastaanottajien sähköpostiosoitteet on häivytetty kuvasta)

Pääkäyttäjän painaessa modaalin lähetyspainiketta vastaanottajien tiedot lähetetään POST-kutsulla palvelintasolle. Tiedot lähetetään JSON-muodossa käyttäen pohjana objektia, joka sisältää taulukon vastaanottajista, sekä tiedon viestin tyypistä (toisin sanoen, onko lähetettävä viesti sisältöpäivitys-kooste vai sähköpostiosoitteen vahvistusviesti) (ks. kuva 23).

```
const data = {
    "users"          : users,
    "message_type"   : message_type
};
```

KUVA 23 Vastaanottajien tiedot ja viestin tyypin sisältävä objekti

5.2.3 Sähköpostien lähetyksen backend-toteutus

Palvelintasolla käyttäjätiedot sisältävä taulukko iteroidaan läpi ja viestien sisällöt koostetaan vastaanottajana olevan käyttäjän ja viestin tyypin mukaan. Osana toiminnallisuuden toteutusta koko QAiRA-projektissa eri kohteissa olleet viestien lähetystoteutukset on nyt korvattu jaetuilla funktioilla tarkoituksena pitää toiminnallisuus yhtenäisenä ja vähentää uuskehityksessä tapahtuvien virheiden määrää. Funktiot hakevat käyttäjätietojen perusteella viestiin vaadittavan datan tietokannasta ja prosessoivat sen sähköpostiviestinä lähetettävään HTML-muotoon.

Esimerkiksi sisältöpäivityskoosteen lähetys on toteutettu funktiolla, joka ottaa parametreinaan tietokantahauissa käytettävän tietokantaobjektin, palvelintasolla vastaanottajan muista tiedoista selvitetyn yksilöllisen tunnisteen, sekä koosteen aikavälin alku- ja loppupäivämäärän (ks. kuva 24). Funktio palauttaa virhe- ja lokitietoja sisältävän taulukon, jota voidaan tarvittaessa käyttää apuna kehityksessä ja virhetilanteiden ratkomisessa.

```
/*
Collects and returns the newsletter product regulation content
@param $db object Database connection
@param $user_hash string Hash of the user
@param $start_date string The earliest date from where to get the product
regulation data (format Y-m-d)
@param $end_date string The last date from where to get the product
regulation data (format Y-m-d)
@return array Product regulation data array defined in the beginning of the
function
*/
function getProductRegulationData($db, $userhash, $start_date, $end_date) {
```

KUVA 24 Sisältöpäivityskoosteen koostavaa ja lähettävää funktiota kuvaileva ohjelmakommentti

Viestin varsinaiseen lähetykseen käytetään jo aiemmin olemassa ollutta jaettua funktiota. Funktio luo Mailgunin PHP-kirjaston mukaisen objektin, joka on rajapinnan kautta yhteydessä Mailgun-järjestelmään, josta viestit varsinaisesti lähetetään vastaanottajille. Kehitettäessä sovellusta paikallisesti sähköpostiviestejä ei lähetetä, vaan viestien koostefunktioihin on laadittu toiminnallisuus, joka sen sijaan kirjoittaa viestin sisällön HTML-tiedostoon, jonka avulla viestin sisältöä ja toiminnallisuuksia voidaan testata, tarkastella ja simuloida.

5.3 Asiakaskäyttäjätietointegraatio

Tietoja QAiRAn asiakasorganisaatioista, -käyttäjistä sekä aktiivisista tilauksista ylläpidetään kahdessa erillisessä kohteessa: QAiRA-sovelluksen omassa sisäisessä tietokannassa sekä Kasven Severa-järjestelmässä. Opinnäytetyön osana toteutetun asiakaskäyttäjätietointegraation tarkoituksena on pitää tiedot ajan tasalla molemmissa järjestelmissä.

Esimerkiksi uuden käyttäjän rekisteröityessä QAiRAan tai asiakkuustilauksen muuttuessa tiedot päivittyvät QAiRAn tietokantaan, mutta eivät muualle. Integraatiotoiminnallisuus vertaa molempien lähteiden tietoja keskenään ja tekee niiden pohjalta tarvittavat muutokset Severa-järjestelmään.

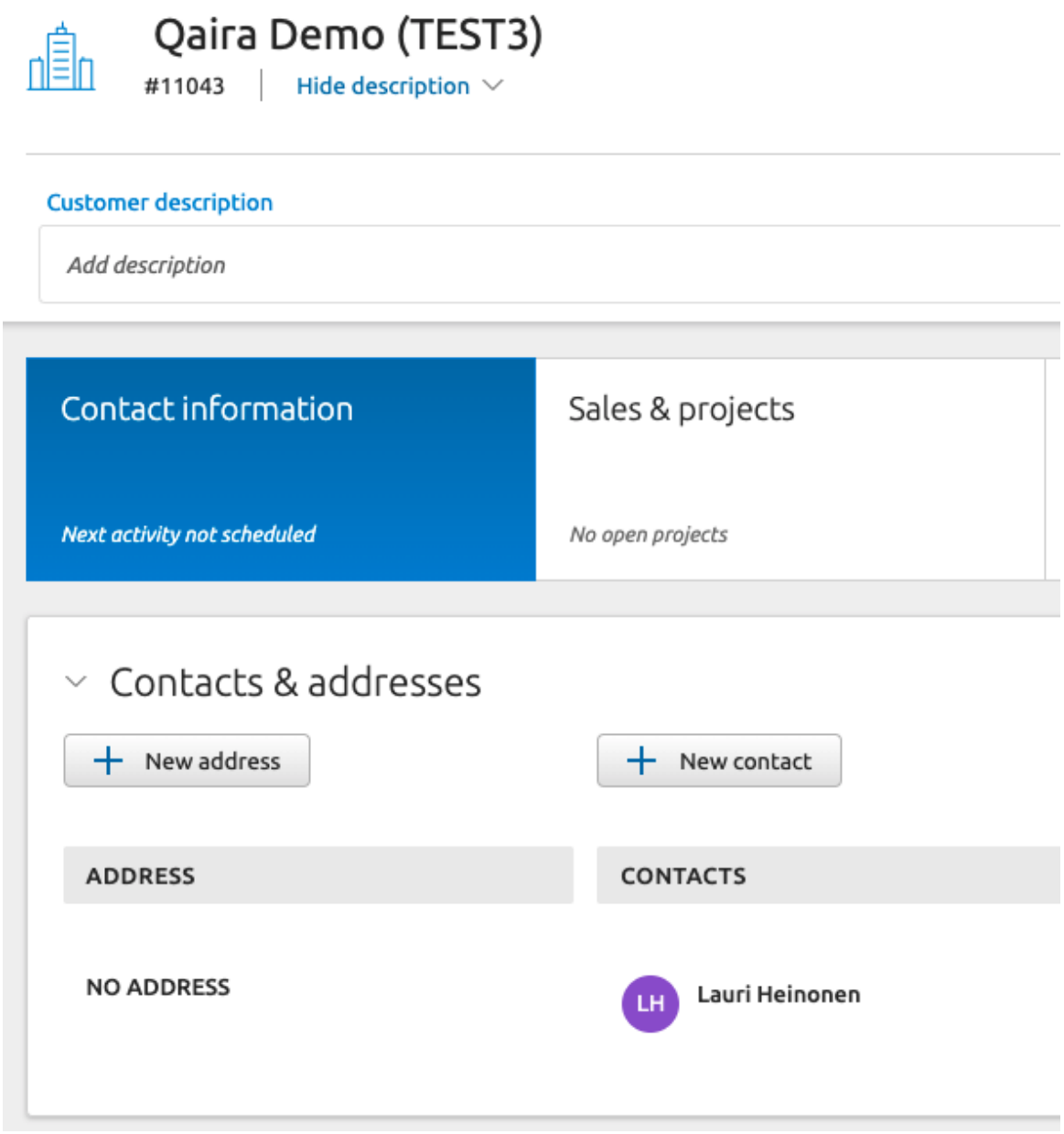
5.3.1 Tietotyypit

Toteutetun integraation kannalta olennaiset tiedot jakautuvat kolmeen eri tietotyyppiin:

1. Asiakasorganisaatiot
2. Henkilökäyttäjät
3. Asiakkuustilauksen tyyppi.

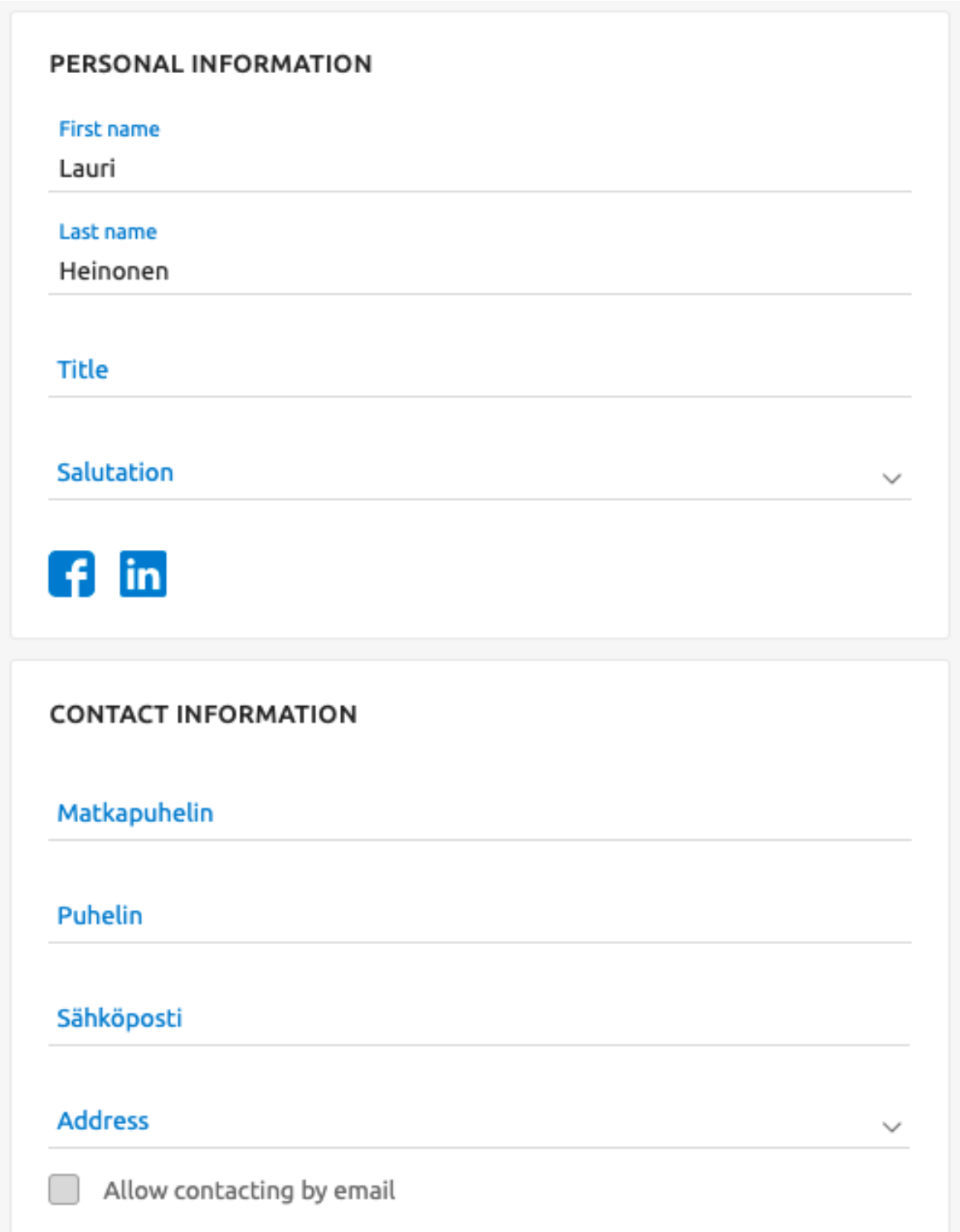
Jokainen tietotyyppi on QAiRAn tietokannassa mallinnettu omaan tauluunsa ja Severan rajapinnassa omaksi tietotyyppiluokakseen.

Asiakasorganisaatiot on rajapinnassa mallinnettu luokalla nimeltä *Customer* (ks. kuva 25). Luokkaan on mahdollista tallentaa muun muassa organisaation nimi, osoite ja Y-tunnus. Kannassa organisaatio on sidottu luokkaan Severa-järjestelmän tuottamalla organisaationumerolla.



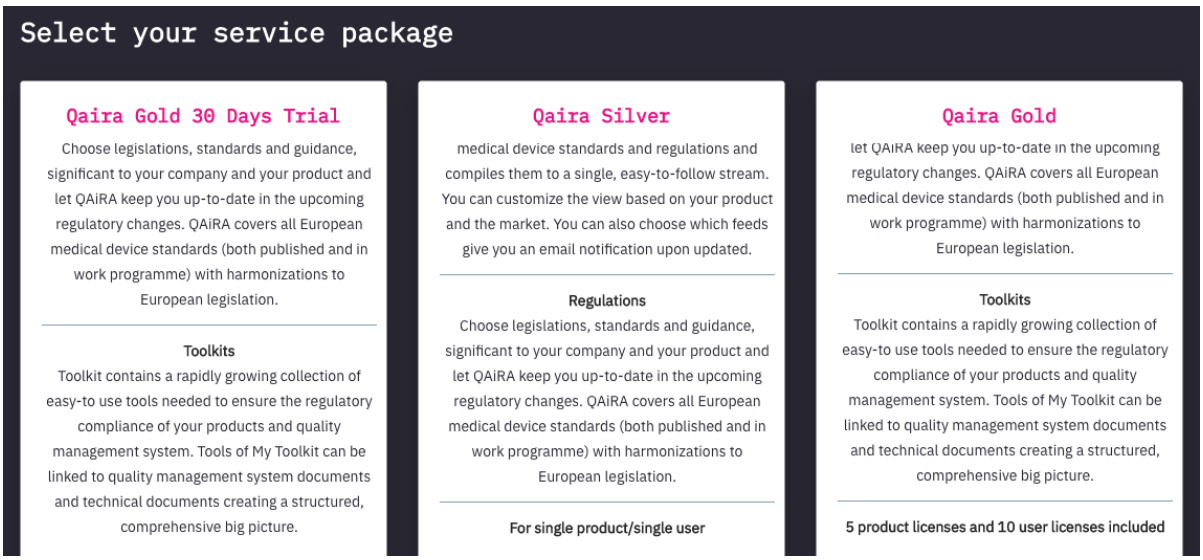
KUVA 25 Yksityiskohta Severan käyttöliittymän asiakasorganisaationäkymästä

Henkilökäyttäjätiedot on tallennettu järjestelmään käyttäen rajapinnassa olevaa luokkaa *Contact*, jonka tarkoituksena on varastoida organisaation yksittäisten työntekijöiden ja henkilöiden yhteystietoja (ks. kuva 26). Toteutuksen kannalta olennaiset käyttäjästä tallennettavat tiedot ovat nimi, sähköpostiosoite ja puhelinnumero. Käyttäjä on sidottu luokkaan käyttäen järjestelmän tuottamaa yksilöllistä id-arvoa.



KUVA 26 Yksityiskohta Severan käyttöliittymän henkilöyhteystietonäkymästä

Asiakkuustilauksen tyyppi tarkoittaa asiakkaan tämänhetkisen QAiRA-tilauksen tyyppiä. Rekisteröityessään QAiRAan käyttäjä valitsee haluamansa tilauksen eli palvelupaketin kolmesta eri vaihtoehdosta (ks. kuva 27). Käyttäjä voi tämän jälkeen muuttaa tilaustaan sovelluksessa voimassa olevin ehdoin. Rekisteröityessään ensimmäistä kertaa käyttäjällä on mahdollisuus valita 30 päivän maksuton kokeilutilaus (*Trial*). Kaksi muuta tilaustyyppiä ovat niin kutsutut *QAiRA Silver* ja toiminnallisuuksiltaan monipuolisin *QAiRA Gold*.

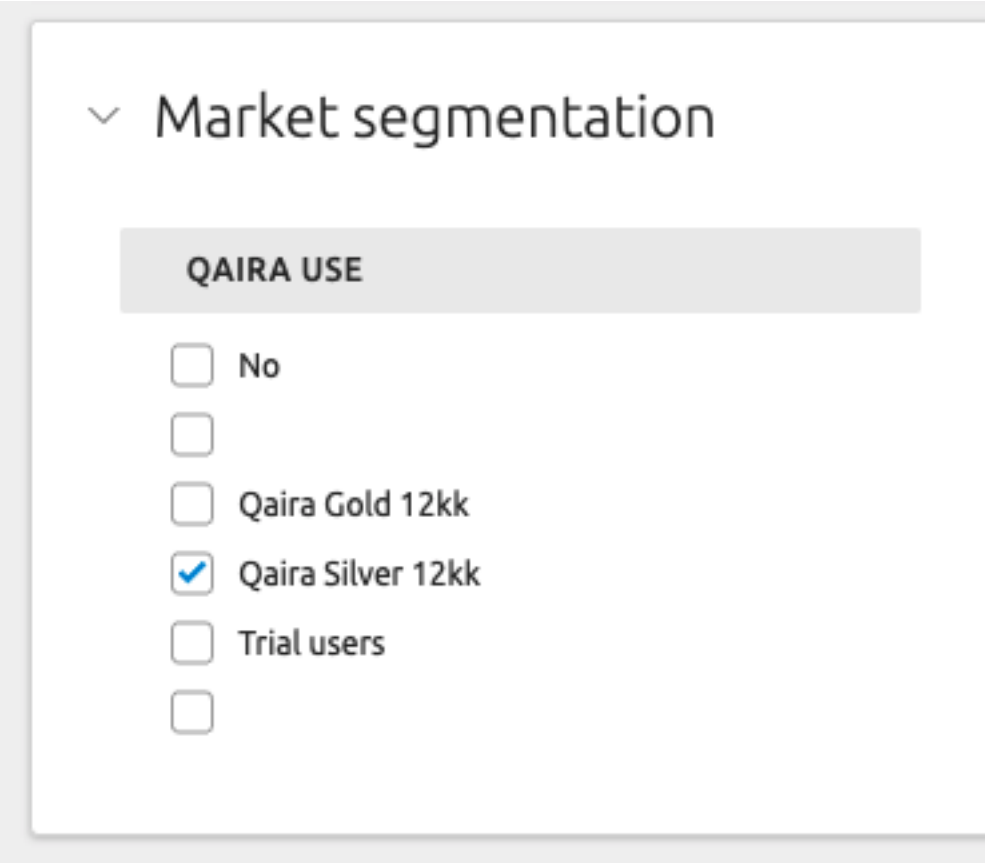


KUVA 27 Yksityiskohta QAiRAn rekisteröitymissivulta, jossa on lueteltu eri asiakkuustilauksien tyypit

Asiakkuustyypit on mallinnettu Severaan käyttämällä rajapintaluokkaa *Market Segment* eli järjestelmän käyttämän käännöksen mukaisesti asiakasryhmä (ks. kuva 28). Järjestelmän käyttöliittymässä asiakasorganisaatio voidaan merkitä QAiRAn käyttäjäksi valitsemalla jokin *QAiRA Use* -nimisen ylätason asiakasryhmäobjektin vaihtoehdoista. Vaihtoehtoja on kuusi, joista integraatiototeutuksen kannalta olennaisia vaihtoehtoja on neljä:

1. No (Ei aktiivista QAiRA-tilausta)
2. QAiRA Gold
3. QAiRA Silver
4. Trial users (Koeajan tilaus).

Jokainen vaihtoehdoista edustaa omaa asiakasryhmäobjektiaan. Myös asiakkuustyypit ovat tietokannassa sidottu luokkaan Severa-järjestelmän tuottamalla yksilöidyllä id-arvolla. Integraation kannalta olennainen tieto on siis asiakasorganisaation ja asiakkuustyypin välinen yhteys, toisin sanoen mikä asiakkuustyyppi on yhdistetty kullekin asiakasorganisaatiolle. Edellä mainittu yhteys on mallinnettu QAiRAn tietokannassa omalla taulullaan ja Severan rajapinnassa omalla CustomerMarketSegment-luokallaan.



KUVA 28 Esimerkki Severan käyttöliittymän Market Segment -näkymästä (Integraation kannalta epäolennaiset vaihtoehdot on häivytetty kuvasta)

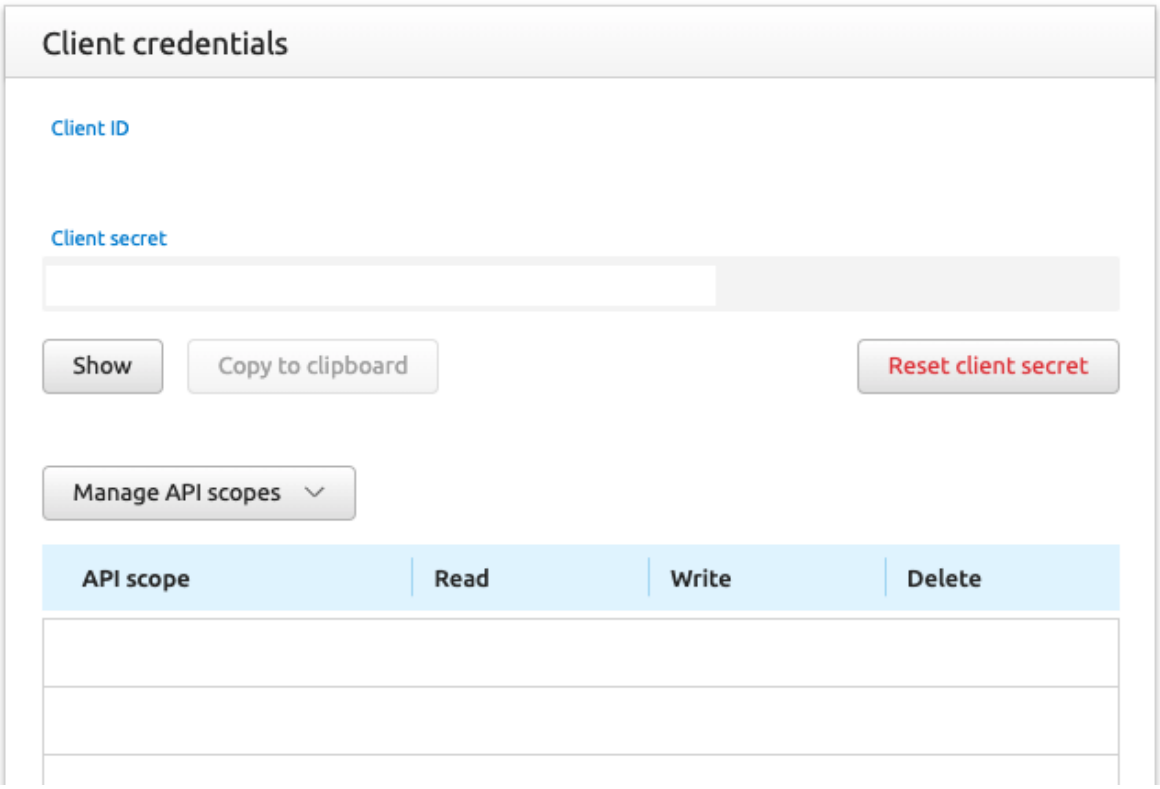
5.3.2 Severa REST-rajapinta

Tietoja voidaan hakea Severasta ja päivittää sinne Visman ylläpitämän REST-rajapinnan kautta. Rajapinta koostuu joukosta verkko-osoitteita, joihin lähetettävien, tarkkojen, ennaltamääritettyjen parametrien ja HTTP-pyyntöjen kautta voidaan olla yhteydessä suoraan järjestelmään. Tietoja vaihdetaan pelkistetyssä JSON-muodossa ilman graafista käyttöliittymää. Rajapinta on siis tarkoitettu lähinnä muiden sovellusten, ohjelmien tai skriptien käytettäväksi.

HTTP-pyynnön tyyppi määrittää, mitä järjestelmän on tarkoitus tehdä. Severan rajapinnassa GET-pyyntöjä käytetään yleisesti tietojen hakuun, POST-pyyntöjä tietojen lisäykseen, PATCH-pyyntöjä muokkaukseen ja DELETE-pyyntöjä tietojen poistoon (Visma REST API). Toteutuksen päivitysskriptit lähettävät HTTP-pyyntöjä PHP:n natiiviversioon kuuluvan CURL-luokkakirjaston avulla.

Severa-rajapinnan käyttö vaatii erillisen rajapintakäyttäjän (ks. kuva 29). Järjestelmä luo käyttäjälle yksilöidyn id:n ja autentikointitunnuksen, joita tarvitaan esimerkiksi skriptiohjelmaa suoritettaessa käytön autentikoinnissa eli käyttöoikeuden todentamisessa. Käytännössä skripti lähettää rajapinnalle ensimmäisenä autentikointipyynnön, jonka mukana lähetetyt id ja tunnus osoittavat pyynnön lähettäjän omaavan kyseisen rajapintakäyttäjän oikeudet. Onnistuneen varmennuksen jälkeen rajapinta lähettää vastauksena autorisointitunnisteen, joka lähettäjän tulee liittää kaikkien tulevien pyyntö-

jensä otsaketietoihin varmentaakseen oikeutensa pyyntöjen toimintoihin. Tunniste on voimassa yhden tunnin ajan (Visma REST API), mikä ei päivitysskriptien suorittamisen kannalta tule käytännössä vastaan, mutta vanhentumisensa jälkeen tunniste on uusittava uudella pyynnöllä.



KUVA 29 Yksityiskohta Severan rajapintakäyttäjien hallinnointinäkymästä (Yksilöivät tiedot on häivytetty kuvasta)

Käyttäjälle voidaan myös määrittää käyttöoikeuksia (scope) eri tietojen luku-, muokkaus- ja poisto-operaatioihin. Käyttäjälle määritettyjä käyttöoikeuksia voidaan suoda yksittäisille rajapintayhteysinstansseille tarpeen mukaan. Tietoturvaa lisäävä suositus on sallia jokaiselle rajapintakäyttäjälle ja -instanssille vain vaadittavat käyttöoikeudet (Visma REST API).

5.3.3 Tietojen päivitystoiminnallisuus

Varsinainen toteutus koostuu QAiRA-järjestelmässä seitsemästä eri tiedostokomponentista. Konfiguraatiotiedostoon on koottu kaikki päivitysprosessissa yleisesti käytetyt vakiot. Funktiotiedostoon on koottu kaikki skriptien käyttämät funktiot. Kolme luokkatiedostoa mallintavat kolmea tietotyyppiä, nimellisesti asiakasorganisaatiota, henkilökäyttäjää ja asiakkuustilausta. Kaksi skriptitiedostoa suorittavat kaksi päivitysprosessin automatisoitua vaihetta.

Päivitysprosessin ensimmäinen vaihe on lähdedatan prosessointi ja päivitysdatan koostaminen. Ensimmäinen päivitysskripti hakee ja jäsentelee asiakkuuksiin liittyvän datan rajapinnan kautta Severasta. Koska Severaan on tallennettu tietoja myös QAiRAan liittymättömistä asiakkuuksista, pitää olennainen data suodattaa asiakasryhmätiedon avulla.

Seuraavaksi skripti hakee ja jäsentelee QAiRAn tietokannassa olevan datan kantahaulla ja vertailee kahta datajoukkoa toisiinsa. Mikäli toisessa datajoukossa oleva tieto puuttuu tai eroaa toiseen datajoukkoon verrattaessa, skripti kirjoittaa tiedot ja tarvittavat muutostoimenpiteet CSV-tiedostoon. Mikäli esimerkiksi QAiRAn tietokannasta löytyvä asiakasorganisaatio puuttuu Severasta, pitää kaikki siihen liittyvät tiedot henkilökäyttäjät ja asiakkuustyyppi mukaan lukien lisätä Severaan. Jos taas Severasta löytyvä asiakasorganisaatio puuttuu tietokannasta, pitää QAiRA-tilauksen tyyppi muuttaa Severassa muotoon ”No” (ei aktiivista tilausta).

Skripti kirjoittaa jokaisen kolmen tietotyypin muutoksille omat CSV-tiedostonsa, joita toinen päivitysskripti käyttää syötteenään. Prosessi on jaettu kahteen erilliseen, automatisoituun vaiheeseen, jotta tietoja voidaan manuaalisesti tarkistaa ja korjata ennen varsinaista päivitystä. Jos CSV-aineistossa havaitaan esimerkiksi organisaatio, jonka tietoja ei haluta lisätä Severaan, voi päivitysprosessin tekijä poistaa kyseisen tietorivin tiedostosta ennen toisen päivitysskriptin suorittamista.

Päivitysprosessin viimeinen vaihe on tarvittavien muutosten tekeminen ja tietojen tallentaminen Severan järjestelmään sekä QAiRAn tietokantaan. Toinen päivitysskripti käy kolmeen CSV-tiedostoon koostetun päivitysdatan läpi rivi kerrallaan ja lähettää tarvittavat muutospyynnöt Severan rajapintaan. Esimerkiksi uuden asiakasorganisaation luotuaan rajapinta palauttaa vastauksena kyseiselle organisaatiolle yksilöidyn id-arvon, jonka päivitysskripti tallentaa QAiRAn tietokantaan.

6 YHTEENVETO JA POHDINTA

Opinnäytetyön tuloksena toteutettiin kolme edellä kuvailtua ominaisuutta, jotka julkaistiin versiopäivitysten yhteydessä sovelluksen tuotantoversioon. Regulaatioaineistopäivitysdatan lisäys sisältöpäivityskoosteeseen on osatoiminnallisuus, joka laajentaa QAiRAn jo olemassa olevia ominaisuuksia. Kaksi muuta kehitettyä ominaisuutta ovat kokonaan uusia. Opinnäytetyöprosessi antoi siis monipuolisesti kokemusta sekä olemassa olevien ominaisuuksien optimoinnista ja koodin refaktoroinnista että uusien ominaisuuksien suunnittelusta ja toteutuksesta. Henkilökohtaisen kehityksen kannalta rajapintojen kanssa tehty työ antoi eniten uutta kokemusta.

Opinnäytetyösuunnitelman mukaisen toteutuksen suurimmaksi hidasteeksi muodostui suunnitelmassa määritelty asiakastietointegraatio, jonka toteutusta siirrettiin tekijästä riippumattomista syistä kehitysprojektissa useaan kertaan myöhemmälle sekä opinnäytetyön että kehitettävän ominaisuuden tärkeyttä painottavista perusteluista huolimatta. Käytännössä muita kehitystehtäviä, kuten muun muassa opinnäytetyössä toteutetut kaksi muuta ominaisuutta priorisoitiin integraatiota tärkeämmiksi ja integraation priorisointijärjestyksestä projektilla oli tekijän itsensä poissulkienkin sisäisiä erimielisyyksiä. Integraatioon suunniteltua aikaa siirrettiin siis muiden ominaisuuksien toteutukseen ja sen laajuus jäi suunniteltua pienemmäksi. Parhaiten projektissa onnistui sähköpostien lähetystoiminnallisuus, jonka määrittely oli kaikista selkein.

Kaikki toteutetut ominaisuudet koettiin kuitenkin hyvin tärkeiksi ja tarpeellisiksi lisäyksiksi sovellukseen. Toteutettujen ominaisuuksien tuottama hyöty jakautui myös hyvin monipuolisesti eri käyttäjäryhmille. Sisältöpäivityskoosteen laajennuksesta hyötyivät sovelluksen loppukäyttäjät, sähköpostien lähetystyökalusta QAiRA-järjestelmäylläpitäjät ja asiakastietointegraatiosta yrityksen sisäiset markkinointi- ja liiketoiminnan hallinto-osastojen käyttäjät.

Jatkokehitysehdotuksina järjestelmäylläpitäjien sähköpostin lähetystyökaluun voisi lisätä käyttäjien haku-, ryhmittely- ja suodatustoiminnallisuuksia, mikä lisäisi ominaisuuden skaalautuvuutta. Käyttäjämäärän kasvaessa suureksi yksittäisen käyttäjän tai organisaatioidenkin paikannus käy nykyisessä toteutuksessa työlääksi. Lisäksi päivämääräkenttään voisi lisätä logiikkaa tai monipuolisemman käyttöliittymäkomponentin. Koska ominaisuus on kehitetty yksinomaan järjestelmäylläpitäjien sisäiseen käyttöön, on toteutus tehty hieman yksinkertaisemmin ajan säästämiseksi.

Asiakastietointegraatioon voisi taas nyt ensimmäisen vaiheen toteutuksen jälkeen tehdä jatkokehityksenä käyttöliittymän, jonka avulla synkronisointi voitaisiin suorittaa jouhevammin ja jonka kautta voitaisiin ideaaleimmillaan päivittää ja hallinnoida myös muita asiakastietoja, kuten esimerkiksi tilausten kestoja. Käyttöliittymän voisi toteuttaa joko QAiRAn sisäisesti tai niin, että tietoja voitaisiin hallinnoida Severan kautta. Edellä mainitut kehitysehdotukset on annettu myös toimeksiantajan käyttöön ja niistä on keskusteltu kehitysryhmän kanssa.

LÄHTEET

Atlassian. Company. Verkkosivu. https://www.atlassian.com/company Viitattu 01.10.2021.

Atlassian. Conflunce. Verkkosivu. https://www.atlassian.com/software/confluence Viitattu 02.10.2021.

Atlassian. Jira. Verkkosivu. https://www.atlassian.com/software/jira Viitattu 02.10.2021.

Carr, David & Gray, Markus 2018. Beginning PHP: master the latest features of PHP 7 and fully embrace modern PHP development. Birmingham: Packt Publishing.

DanielSHaischt 2013. Kuva. https://en.wikipedia.org/wiki/Ajax_(programming)#/media/File:Ajax-vergleich-en.svg Viitattu 27.09.2021.

Euroopan komissio. Kansanterveys: Medical Devices – Sector. Verkkosivu. https://ec.europa.eu/health/md_sector/overview_fi Viitattu 12.11.2021.

Geisshirt, Kenneth, Olsson, Aske, Voss, Rasmus & Zattin, Emanuele 2018. Git Version Control Cookbook: Leverage Version Control to Transform Your Development Workflow and Boost Productivity. 2. painos. Birmingham: Packt Publishing.

Git. Verkkosivu. https://git-scm.com/ Viitattu 02.10.2021.

Git. A Short History of Git. Verkkosivu. https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git Viitattu 02.10.2021.

Git. About. Verkkosivu. https://git-scm.com/about Viitattu 02.10.2021.

GitHub. Pricing. Verkkosivu. https://github.com/pricing Viitattu 03.10.2021.

GitLab. About GitLab. Verkkosivu. https://about.gitlab.com/company/#the-company Viitattu 03.10.2021.

Harned, David 2018. Hands-On Agile Software Development with JIRA: Design and Manage Software Projects Using the Agile Methodology. Birmingham: Packt Publishing.

Haverbeke, Marijn 2019. Eloquent Javascript: a modern introduction to programming. 3. painos. San Francisco: No Starch Press.

Hethey, Jonathan 2013. GitLab Repository Management. Birmingham: Packt Publishing.

Kasve. Yritys. Verkkosivu. https://kasve.com/yritys/ Viitattu 08.11.2021.

Kasve. QAiRA. Verkkosivu. https://kasve.com/qaira/ Viitattu 11.11.2021.

Mailgun. Company / About. Verkkosivu. https://www.mailgun.com/company/ Viitattu 30.09.2021.

Mailgun. Documentation / Introduction. Verkkosivu. https://documentation.mailgun.com/en/latest/api-intro.html Viitattu 30.09.2021.

MariaDB. About. Verkkosivu. https://mariadb.org/about/ Viitattu 29.09.2021.

MariaDB. Knowledge Base, Why is the Software Called MariaDB? Verkkosivu. https://mariadb.com/kb/en/why-is-the-software-called-mariadb/ Viitattu 29.09.2021.

OpenJS Foundation. Verkkosivu. https://openjsf.org/ Viitattu 27.09.2021.

PHP. Credits. Verkkosivu. https://www.php.net/credits.php Viitattu 25.09.2021.

PHP. History of PHP. Verkkosivu. https://www.php.net/manual/en/history.php.php Viitattu 23.09.2021.

PHP. License Information. Verkkosivu. https://www.php.net/license/index.php Viitattu 25.09.2021.

PHP. Preface. Verkkosivu. https://www.php.net/manual/en/preface.php Viitattu 25.09.2021.

Sriparasa, Sai Srinivas. 2014. Building a Web Application with PHP and MariaDB: A Reference Guide. Birmingham: Packt Publishing.

Visma. Tietoa Vismasta. Verkkosivu. https://www.visma.fi/tietoa-vismasta/ Viitattu 30.09.2021.

Visma. Visma Severa. Verkkosivu. https://psa.visma.fi/ Viitattu 30.09.2021.

Visma. Visma Severa API Documentation. Verkkosivu. https://api.severa.visma.com/ Viitattu 30.09.2021.

Visma REST API. REST API Guide. Verkkosivu. https://api.severa.visma.com/rest-api-doc Viitattu 04.11.2021.

Visma REST API. Visma Severa Public Rest API Documentation. Verkkosivu. https://api.severa.visma.com/rest-api/doc/index.html Viitattu 04.11.2021.

W3Techs. Historical trends in the usage statistics of server-side programming languages for websites. Verkkosivu. https://w3techs.com/technologies/history_overview/programming_language Viitattu 25.09.2021.

York, Richard 2015. Web Development with jQuery. 2. painos. Indianapolis: John Wiley & Sons, Inc.