Sami Kurvinen

Tietoturvallinen kertakirjautumisjärjestelmä keskisuurelle organisaatiolle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

18.3.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Sami Kurvinen Tietoturvallinen kertakirjautumisjärjestelmä keskisuurelle organisaatiolle 47 sivua 18.3.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Hallituksen puheenjohtaja Jarmo Keto Yliopettaja Kari Aaltonen

Insinöörityön tarkoituksena oli kehittää keskisuurelle organisaatiolle tietoturvallinen kertakirjautumisjärjestelmä. Organisaatiolla oli monta paikallisen käyttäjähallinnan sisältävää verkkopalvelua, joiden hallinta koettiin työlääksi. Kertakirjautumisjärjestelmän tavoitteena oli helpottaa käyttäjähallintaa, luoda tietoturvaa ja lisätä organisaation verkkopalveluiden käyttöönottoa.

Kertakirjautumisjärjestelmä kehitettiin kahden vuoden aikana projektiluontoisesti. Projekteissa laajennettiin kertakirjautumisjärjestelmän toimintoja ja liitettiin organisaation eri verkkopalveluja sen piiriin. Jokaisen projektin aikana varmennettiin, että järjestelmä on kehittymässä oikeaan suuntaan, keräämällä palautetta sen käytöstä.

Insinöörityön tuloksena organisaatiolle luotiin verkkopalveluna toimiva kertakirjautumisjärjestelmä, yleiskäyttöinen koodikirjasto ja valmiita liitännäisiä muiden verkkopalvelujen liittämiseksi kertakirjautumisjärjestelmään. Kertakirjautumisjärjestelmässä hallitaan henkilöiden käyttäjätunnuksia, käyttöoikeusryhmiä ja järjestelmän piirissä olevia sovelluksia. Koodikirjastolla kommunikoidaan kertakirjautumisjärjestelmän kanssa ja tehdään käyttäjän tunnistautumisia. Liitännäisillä otetaan kertakirjautumisjärjestelmä käyttöön Wordpress-sivustoissa tai Simple Machines Forum -keskustelupalstoilla.

Kertakirjautumisjärjestelmä kehitettiin PHP-ohjelmointikielellä Laravel-ohjelmistokehystä käyttäen. Järjestelmien välinen kommunikointi ja tunnistautuminen toteutettiin REST-rajapinnoilla. Kertakirjautumisjärjestelmään liitetyt sovellukset ja niiden tekemät kyselyt tunnistettiin RSA-avainpareilla.

Palautteen perusteella kertakirjautumisjärjestelmä on saavuttanut sille asetetut tavoitteet. Organisaation henkilöstö koki käyttäjähallinnan helpottuneen, ja vekkopalveluiden käyttöönotto on lisääntynyt. Kertakirjautumisjärjestelmässä ei ole havaittu tietoturvapuutteita, joista olisi koitunut haittaa järjestelmän käyttäjille tai organisaatiolle.

Kertakirjautumisjärjestelmä jää organisaation omaisuudeksi ja jatkokehitettäväksi. Insinöörityön raportti sisältää käyttökelpoista tietoa keskitetyn verkkopalvelun suunnittelijalle ja kehittäjälle.

Avainsanat	tietoturva, kertakirjautumisjärjestelmä, käyttäjähallinta

Author Title Number of Pages Date	Sami Kurvinen Secure single sign-on system for small to medium organization 47 pages 18 March 2016
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	Digital Media
Instructors	Jarmo Keto, Chairman of the Board Kari Aaltonen, Principal Lecturer

Goal of this project was to develop a secure single sign-on system for a small to medium sized organization. The organization had multiple web services with local user account management, which was considered to be laborious. The goal for the single-sign on system was to ease the user account management of the organization across all web services, create information security and increase the web service adoption with the personnel.

The single sign-on system was developed across two years in project based iterations. The projects extended the system's features and new web services were incorporated into the single sign-on system. The direction of the development and the service as a whole were validated by collecting feedback on the usage of the service.

The project resulted in a working web based single sign-on system, a general code library and extensions to other web applications for incorporating the single sign-on system into them. The single sign-on system manages the organization's user accounts, access groups and applications in the system. The code library is used to communicate with the system and to handle the user authentication. Extensions are used to integrate the single sign-on system into existing web applications, such as Wordpress or Simple Machines Forum.

The single sign-on system was developed with the PHP programming language, using the Laravel framework. Communication between the single sign-on system and applications in its scope follows the REST-API convention. All communications were identified with the RSA-public key infrastructure.

The single sign-on system reached all its goals based on feedback. The organization's personnel felt that user account management have become easier and that web services are now adopted faster and more efficiently than before. There have been no reports of security faults that would have affected the system's users or the organization.

The single sign-on system will remain as the organization's intellectual property and is theirs to continue development. This thesis contains valuable information for people designing or developing a centralized web service.

Keywords	Information security, Single Sign-On, User Management

Sisällys

Lyhenteet

Bcrypt	Niels Provosin ja David Mazièresin suunnittelema vahva salausalgoritmi, joka perustuu Blowfish-salausalgoritmiin.
CSRF	Cross Site Request Forgery. Tietoturvahaavoittuvuus, jossa hyödynnetään palvelimen luottamusta käyttäjään ja tehdään luvattomia verkkokyselyjä käyttäjän nimissä.
DELETE	HTTP-protokollan mukainen metodi, jolla poistetaan resurssi.
GET	HTTP-protokollan mukainen metodi, jolla ladataan verkkosivu tai resurssi.
Git	Hajautettu versionhallintajärjestelmä.
HTTP	Hyper Text Transfer Protocol. Verkkosivujen tiedonsiirtoon käytetty protokolla.
Kanban	Projektinhallintamalli tai työkalu, joka seuraa tehtävien tilaa prosesseina. Saanut alkunsa Japanista Toyotan tehtaalta.
MD5	Message Digest. Murrettu tiivistealgoritmi, jonka käyttöä ei suositella.
MVC	Model View Controller. Ohjelmistomalli, jolla eriytetään eri ohjelmakoodien vastuualueita.
MySQL	Relaatiotietokanta, jossa tiedolle annetaan tarkasti määritetty rakenne ja viittauksia toisiin tietueisiin.
NSA	National Security Agency. Yhdysvaltain kansallinen turvallisuusvirasto. Erikoistunut kryptografiaan ja signaalitiedusteluun.
NoSQL	Not only SQL. Tietokantatoteutus, joka ei ole relaatiopohjainen.
OpenSSL	Open Secure Sockets Layer. Avoimen lähdekoodin työkalupakki TLS- ja SSL-protokollien hyödyntämiseen.

PHP	PHP: Hypertext Preprocessor. Suosittu skriptikieli verkkopalveluissa.
POST	HTTP-protokollan mukainen metodi, jolla lähetetään tietoa verkkosivuille tai luodaan resurssi.
PUT	HTTP-protokollan mukainen metodi, jolla tallennetaan resurssi.
PRISM	Planning Tool for Resource Integration, Synchronization, and Management. NSA:n vakoiluohjelma, joka tekee yhteistyötä Yhdysvaltojen suuryritysten kanssa.
REST	Representational State Transfer. Rajapintamalli, joka perustuu verkko-osoitteen osoittamiin resursseihin ja HTTP-metodeihin.
RSA	Julkisen avaimen salausalgoritmi. Nimi tulee kehittäjien sukunimien ensimmäisistä kirjaimista.
Scrum	Projektinhallintamalli, joka on suosittu ketterässä ohjelmistokehityksessä. Toiminta perustuu lyhyisiin sykleihin, joita kutsutaan sprinteiksi.
XML	Extensible Markup Language. Yläkäsite merkintäkielille, joissa voidaan määrittää tiedon rakenne ja merkitys.
XSS	Cross Site Scripting. Tietoturvahaavoittuvuus, jossa hyödynnetään käyttäjän luottamusta verkkosivuun ja suoritetaan ohjelmakoodia käyttäjän selaimessa.

1 Johdanto

Suurten ja keskisuurten organisaatioiden henkilöstön käytössä olevien verkkopalveluiden käyttöoikeushallinta muodostuu usein monivaiheiseksi ja hankalaksi. Tämän insinöörityön tavoitteena on vähentää keskisuuren organisaation käyttöoikeushallinnasta koituvaa työtaakkaa, korottaa käyttäjätunnuksiin liittyvää tietoturvaa ja lisätä organisaation henkilöstön verkkopalveluiden käyttöönottoa kehittämällä tietoturvallinen kertakirjautumisjärjestelmä.

Suuri osa yritysten ja organisaatioiden toiminnasta on internetissä erilaisten verkkopalveluiden muodossa. Monella organisaatiolla on useita verkkopalveluita niin sisäisessä kuin ulkoisessakin käytössä. Näiden palveluiden käyttöoikeuksien hallinta paikallisesti karkaa helposti organisaatioiden käsistä, ja ne muodostuvat varsin työläiksi hallinnoida, mikä aiheuttaa aikataulu- ja kustannusrasitteita. Tämä ongelma ratkaistaan usein ottamalla käyttöön jonkinlainen keskitetty käyttäjähallinta, kuten kertakirjautumisjärjestelmä.

Tässä insinöörityössä suunnitellaan ja kehitetään tietoturvallinen kertakirjautumisjärjestelmä verkkopalveluna. Järjestelmään voidaan liittää asiakassovelluksia, jotka ottavat kertakirjautumisjärjestelmän käyttöön siihen kehitettävien suljettujen rajapintojen avulla. Järjestelmä varmentaa asiakassovelluksien aitouden. Järjestelmässä voidaan hallita käyttäjien tietoja, käyttöoikeuksia ja keskinäisiä hierarkioita asettamalla käyttäjiä ryhmiin. Kaikki järjestelmässä tehdyt toimet voidaan todentaa ja kohdentaa niistä jäävien merkintöjen avulla asiaankuuluviin henkilöihin ja kellonaikoihin.

Järjestelmän on tilannut keskisuuri organisaatio helsinkiläiseltä digitoimistolta, G-Works Oy:ltä vuoden 2013 keväällä. G-Works Oy on noin 10 henkeä työllistävä digitaalisiin ratkaisuihin keskittyvä yritys. Tilaaja ei ole antanut lupaa nimensä julkaisuun. Toimin G-Works Oy:n palveluksessa vuosina 2012–2015 järjestelmäasiantuntijana. Kertakirjautumisjärjestelmää kehittäessäni toimin projektin teknisenä suunnittelijana ja toteuttajana.

2 Tietoturva ja kertakirjautumisjärjestelmät yleisesti

2.1 Tietoturvan määritelmä

Tietoturva muodostuu tiedon luottamuksellisuuden, eheyden ja saatavuuden takaamisesta (Rousku 2014: 47). Tämä pätee tietojärjestelmien ja verkkopalveluiden sekä fyysisten dokumenttien tietoturvaan.

Luottamuksellisuudella pyritään takaamaan, että luottamuksellinen tieto pysyy vain siihen oikeutettujen henkilöiden ja tahojen saatavilla. Tieto ei siis saa joutua ulkopuolisten käsiin minkään tietomurron, varkauden tai inhimillisen virheen seurauksena. Eheydellä tarkoitetaan tiedon oikeellisuuden takaamista. Tieto ei saa muuttua hallitsemattomasti sellaisten tahojen toimesta, joilla ei ole siihen oikeutta, eikä vääristyä sitä esitettäessä. Saatavuudella tarkoitetaan, että tieto on ylipäätänsä saatavilla silloin, kun sitä tarvitaan, ja riittävän helposti. Verkkopalveluissa se tarkoittaa, että palvelun täytyy olla käynnissä ja vasteaikojen riittävän pieniä.

Chris Shiflettin PHP Security Guide -ohjeessa todetaan, että tietoturva on mitattava määre. Se koostuu useista tietoturvaan liittyvistä kysymyksistä ja niiden mittauksista. Näiden määreiden listaaminen puhekielessä olisi kuitenkin hyvin epäkäytännöllistä, joten on luontevampaa kuvailla järjestelmää tietoturvalliseksi tai -turvattomaksi. Tietoturva terminä on hyvin subjektiivinen: Villen Nakkikioskin tietoturvallinen internetsivu tarkoittaa hyvin eri asiaa kuin Osuuspankin tietoturvallinen verkkopankki. Nakkikioskin riskinä voi olla yksi menetetty työpäivä ja huono maine tuhratuista verkkosivuista, kun taas verkkopankin riskit ovat taloudellisesti hyvin merkittäviä.

Kun halutaan määrittää oman verkkopalvelun tietoturvan taso, se täytyy tasapainottaa kustannusten kanssa. Pienellä työmäärällä saadaan kasvatettua tietoturvaa, mutta tietyn pisteen jälkeen kustannukset nousevat merkittävästi ja ne täytyy suhteuttaa niistä saataviin hyötyihin ja mahdollisiin menetyksiin.

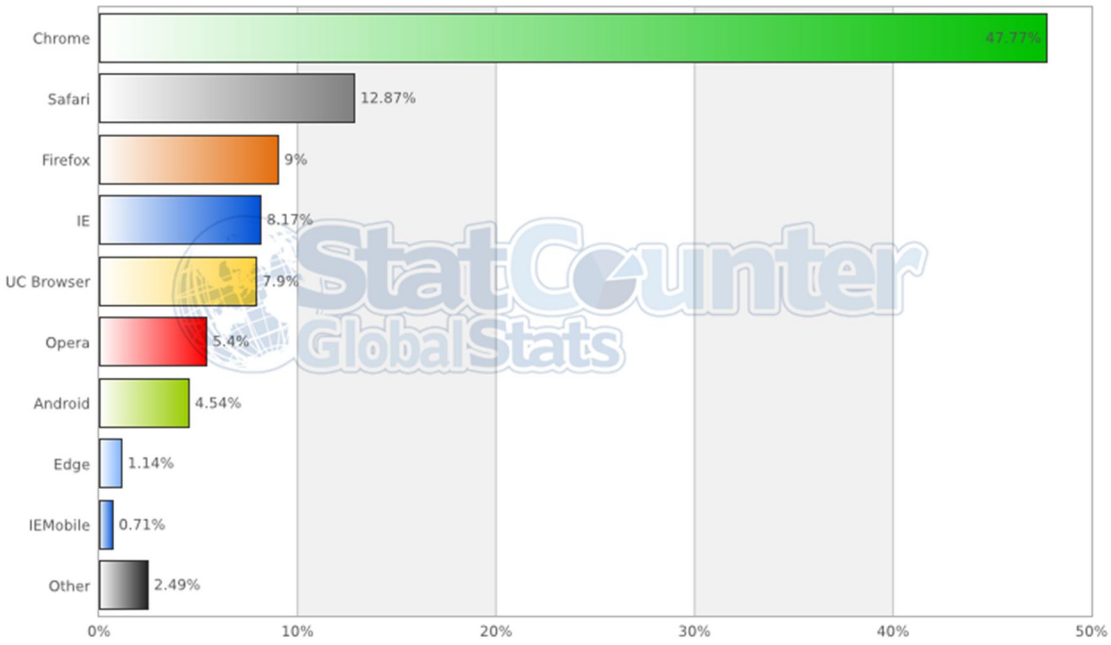
Käytettävyys ja käyttökokemus ovat yhä useammin kriittisessä roolissa uusia verkkosivuja ja -palveluita perustettaessa. Ne ovat usein vastakkain tietoturvan kanssa, ja niiden välille täytyy saada riittävä tasapaino. Käyttäjälle olisi kaikkein miellyttävintä, jos hän voisi

vain mennä päivittämään verkkosivustoaan välittämättä käyttöoikeuksista tai salasanoista. Niiden tuoma käyttäjärajaus on kuitenkin usein tärkeämpää kuin se pieni käyttömukavuus, joka saavutetaan jättämällä ne pois.

Tietoturvan mitattavat määreet liittyvät usein käyttäjän pääsyyn tiettyyn tietoon tai kykyyn tehdä palvelulla jotain, mitä hän ei saisi tehdä, esimerkiksi "Voiko verkkosivun käyttäjä lukea tiedotteita" tai "Voiko tunnistamaton käyttäjä muokata tiedotetta". Riittävästi vastaavanlaisia kysymyksiä kysymällä saadaan kartoitettua verkkosivun tai -palvelun tietoturvan taso.

2.2 Verkkopalveluiden tietoturva

Verkkopalveluiden erikoisuus perinteisiin tietokonesovelluksiin nähden on niiden ympäristö. Siinä missä sovellusta käytetään kehittäjien luomilla työkaluilla, verkkopalveluja selataan useilla eri selaimilla ja niiden käyttäjiä on vaikea ennustaa. Verkkopalveluiden käyttäjät muodostavatkin suuren puutteen tietoturvaan (Zalewski 2011). Käyttäjät itsessään eivät osaa suojautua verkossa, mikä voi johtaa esimerkiksi vahingollisten linkkien klikkailuun ja tietojen kalasteluun. Kuvassa 1 nähdään vuoden 2016 helmikuun tilasto suosituimmista verkkoselaimista.



Kuva 1. Helmikuun 2016 suosituimmat selaimet StatCounter Global Statsin mukaan (StatCounter Global Stats: 2016).

Eri selaimia ja niiden eri versioita on siis paljon, ja jokaisella käyttäjällä on vielä oma asennuksensa kyseisestä selaimesta. Tämä on yksi syistä, jotka ovat johtaneet verkkopalveluiden tietoturvaa ohjanneeseen toteamukseen ”kaikki käyttäjäsyöte on epäluotettavaa” (Stuttard & Pinto 2011: 17). Käytännössä tämä tarkoittaa, että kaikki verkkopalvelulle tulevat syötteet tulee varmentaa tai muokata turvalliseksi eikä päästää sellaisenaan käsittelyyn. Käyttäjä voi tahallaan tai tahattomasti lähettää sellaista sisältöä syötteessään, joka on käsittelemättömänä vahingollista verkkopalvelulle.

Maailmanlaajuinen voittoa tavoittelematon järjestö The Open Web Application Security Project (OWASP) seuraa ja kehittää verkkosovellusten tietoturvaa. OWASP listaa 10 yleisintä verkkopalveluiden tietoturvahaavoittuvuutta, joista kolme yleisintä ovat seuraavat:

1. Injektiohaavoittuvuudet eli haavoittuvuudet, joissa suoritettavan ohjelmakoodin sekaan saadaan ujutettua varmentamatonta syötettä. Nämä usein kohdistuvat tietokantakyselyihin, jolloin vaikutukset osuvat jokaiseen kolmesta tietoturvan pilarista: tieto ei ole luottamuksellista, kun se voidaan injektiolla hakea ohittaen käyttöoikeustarkistukset, se ei ole eheää, kun sitä voidaan luvatta muokata, eikä se ole saatavilla, jos se luvatta poistetaan.

2. Rikkonainen tunnistautuminen tai istuntojen hallinta eli haavoittuvuudet, joissa käyttöoikeuksien tarkistus tai käyttäjän istuntojen hallinta on toteutettu puutteellisesti. Käytännössä näitä haavoittuvuuksia hyödyntämällä voidaan tehdä toimia toisen käyttäjän nimissä ja rikkoa luottamuksellisuus ja eheys riippuen saavutetuista käyttöoikeuksista.

3. Vierasperäisen ohjelmakoodin suorittamiseen (Cross site scripting, XSS) liittyvät haavoittuvuudet eli vierasta ohjelmakoodia päätyy verkkoselaimen suoritettavaksi. Tällä voidaan hyödyntää muita mahdollisia haavoittuvuuksia, kuten kaapata istuntoja tai vaikka näyttää yrityksen verkkosivuilla imagoa tahrivaa sisältöä, mikä rikkoo tiedon eheyttä. (OWASP Top 10 - 2013: 3).

Verkkopalveluissa liikkuu merkittävä määrä tietoa verkon välityksellä, ja usein se halutaan salata. Kryptografialla tarkoitetaan tiedettä, jossa voidaan salata tietoa siten, että se ei ole ulkopuoliselle luettavissa (Kern, Kesavan & Daswani 2007: 203). Tietoa voidaan

salata yksisuuntaisesti, siten että kun tieto on kerran salattu, sitä ei pysty enää kääntämään takaisin alkuperäiseen muotoonsa. Tietoa voidaan myös salata kaksisuuntaisesti, jolloin salattu tieto voidaan palauttaa alkuperäiseen muotoonsa käyttämällä jotain salausavainta, jonka perusteella salaus voidaan purkaa.

Verkkopalveluissa kryptografiaa käytetään hyvin laajasti. Yksi syistä käyttää kryptografiaa on yksityisen ja aran tiedon, erityisesti salasanojen, salaaminen. Jos kaikki tietoturvakeinot pettävät ja vahingollinen käyttäjä pääseekin lukemaan esimerkiksi käyttäjätietokantaa, siellä on syytä säilyttää vain yksisuuntaisesti salattuja salasanoja. Näin vahingollinen käyttäjä ei saa muiden käyttäjien salasanoja tietoonsa selkokielisenä. Hyvä salaustapa on esimerkiksi bcrypt-algoritmi, joka on yleisesti tunnistettu turvalliseksi. Aikaisemmin suosiossa ollut md5-algoritmi on onnistuttu purkamaan, eikä se ole enää tietoturvallinen (Stevens 2007: 42).

Kaikki salaukset kannattaa niin sanotusti suolata, eli salatun tiedon sekaan syötetään jokin satunnainen merkkijono. Näin vältytään hyökkäyksiltä, joissa vahingollisella käyttäjällä on entuudestaan tiedossaan satoja tuhansia teksti-salauspareja, jolloin hän voi suoraan yhdistää salatun tekstin salaamattomaan tekstiin. Salatut tiedot varmennetaan salaamalla aina esimerkiksi syötetty salasana uudestaan ja vertaamalla sitä jo entuudestaan salattuun salasanaan tietokannassa. Mikäli salatut tiedot eivät täsmää, on salasana väärin.

2.3 Tietoturvan todentaminen

Tietoturva ei ole koskaan kuvaava termi verkkosivulle tai -palvelulle. Sen täytyy olla aina mitattavissa oleva määre, ja jotta voidaan luottavaisesti sanoa, että verkkopalvelu on tietoturvallinen, se täytyy pystyä todentamaan.

Paco Hope ja Ben Walther (2008: 2) toteavat teoksessaan, että tietoturvallisuuden testaus ja varmentaminen ovat konkreettisten todisteiden esittämistä siitä, että asetetut turvallisuusmääritykset täyttyvät, vaikka käyttäjä pyrkisi aktiivisesti kiertämään niitä.

Verkkopalvelun kehittäjien tulee ohjelmointityötä tehdessään tehdä kattavaa testausta, jotta he voivat sanoa ohjelman toimivan, kuten on määritelty. Tämän testauksen yhtey-

dessä tulee kokeilla kunkin toiminnallisuuden tietoturvaa pyrkimällä aktiivisesti kiertämään kehittämänsä rajoitteet. Testauksesta tulee dokumentoida, mitä on testattu, miten sitä on testattu ja miten verkkopalvelu on toiminut testin aikana. Dokumentaation avulla voidaan todentaa tietoturvan taso. Verkkopalvelua on myös mahdollisuuksien mukaan hyvä testauttaa ulkopuolisilla henkilöillä, jotta voidaan arvioida tietoturvatoimien vaikutuksia käytettävyyteen.

Kun verkkopalvelun kehittäjät ovat itse tyytyväisiä todentamaansa tietoturvan tasoon, he voivat vielä tilata verkkopalveluunsa ulkopuolisen tietoturva-auditoinnin. Tietoturva-auditoinnissa pätevöitynyt ulkopuolinen taho suorittaa seikkaperäisen tietoturvatestauksen, luo siitä testausraportin ja myöntää sen perusteella yleensä jonkin arvosanan tai sertifikaatin.

Auditointi luo uskottavuutta, kun ulkopuolinen taho on todennut tietoturvan olevan tietyllä tasolla. Verkkopalvelu ei välttämättä läpäise auditointia ensimmäisellä kerralla, mutta siitä saatavan raportin perusteella on helppo kehittää sitä oikeaan suuntaan ja tilata uusi auditointi.

Verkkopalveluiden tietoturvaloukkauksissa tai vikatilanteissa usein nähdään vain aiheutunut haitta. Verkkopalvelu saattaa olla kokonaan tavoittamattomissa, tai osa sitä ei välttämättä toimi. "Mitä on tapahtunut?" on yleensä ensimmäinen kysymys vastaavassa tilanteessa, ja lokikirjan pitäminen on tapa vastata tähän kysymykseen.

Lokikirjaan määritetään, mitä verkkosivun tapahtumia halutaan seurata, ja sen jälkeen lokikirja täyttyy merkinnöistä sitä mukaa, kuin tapahtumia tapahtuu. Lokikirjan avulla voidaan esimerkiksi todentaa, milloin mitäkin verkkopalvelun osaa on käytetty, kuka sitä on käyttänyt ja miten.

Lokikirjasta paljastuu usein esimerkiksi laajamittaisia hyökkäyksiä, kun jokin taho etsii verkkopalvelusta eri osoitteita, joissa voisi olla tietoturva-aukko. Lokikirjan avulla voidaan usein todentaa, onko tietoturva riskeerattu vai onko mahdollisia hyökkäyksiä onnistuttu estämään.

2.4 Tunnistautuminen

Tunnistautumisella tarkoitetaan käyttäjän tunnistamista tietyksi henkilöksi. Kun käyttäjä on tunnistettu, voi verkkopalvelu tarjota räätälöityä sisältöä juuri kyseiselle tunnistetulle henkilölle. Tunnistautumistapoja on monia erilaisia, esimerkiksi ovimies tunnistaa työntekijän kasvoista ja päästää ovesta sisälle. Esimies voi soittaa puhelimitse työntekijälle ja pyytää häntä tekemään jotain – näin esimies tunnistautuu äänellä.

Verkkopalveluissa kuitenkin tunnistautuminen tapahtuu perinteisemmin käyttäjätunnuksella ja salasanalla. Tunnistautumisesta tai autentikoinnista puhuttaessa ei kuitenkaan tule sekoittaa sitä autorisointiin, jolla tarkoitetaan käyttöoikeuksien tarkistusta. On siis eri asia tunnistaa käyttäjä tai rajata hänen käyttöoikeuksiaan.

Kaikessa tunnistautumisessa on lähtökohtaisesti kyse luottamuksesta. Kun henkilö tai järjestelmä tunnistaa käyttäjän, hän luottaa saamaansa informaatioon ja on sen pohjalta vakuuttunut, että kyseessä on oikea henkilö. On kuitenkin syytä tarkastella saatua informaatiota kriittisesti: voiko puhelimessa olla vain hyvin samankaltainen ääni kuin pomolla tai ovatko annetut tunnisteet riittävät. Tämä luottamuksen taso vaihtelee tilanteesta ja ympäristöstä riippuen paljonkin, ja se täytyy määrittää tarkkaan, jotta voidaan sanoa, että järjestelmä pystyy tunnistamaan käyttäjän riittävän tarkasti.

Esimerkkinä luottamuksen tasosta voidaan esittää pienyrityksen kotisivut ja verrata niitä esimerkiksi verkkopankkiin. Pienyrityksen toimitusjohtaja voi mahdollisesti soittaa palveluntarjoajalleen ja tunnistaa itsensä puhelimitse nimellään ja pyytää pienen sisältöpäivityksen sivuillensa ilman ongelmaa. Sama toimitusjohtaja ei kuitenkaan voi soittaa pankkiin ja pyytää pientä tilisiirtoa, vaan hänen on todennettava henkilöllisyytensä tarkemmin, tässä tapauksessa omilla verkkopankkitunnuksilla.

Verkkopankkitunnukset ovat luotettava tapa todentaa henkilöllisyys, sillä niitä varten on täytynyt näyttää henkilökohtaisesti virallinen henkilöllisyystodistus. Verkkosivujen palveluntarjoaja voisi myös pyytää toimitusjohtajaa tunnistamaan itsensä verkkopankkitunnuksilla, mutta se olisi käytännön tarkoituksiin liian kömpelöä ja vaivalloista. Riskit ovat myös usein merkittävästi pienempiä. Riittävä luottamuksen taso määritelläänkin usein mahdollisen riskin kautta: mikäli virheellisellä tunnistautumisella voi saada merkittävää haittaa aikaiseksi, on perusteltua vaatia luotettavampi tunnistautuminen.

2.5 Kertakirjautumisjärjestelmät

Verkkopalveluihin tai muihin järjestelmiin täytyy usein kirjautua sisään eli tunnistautua joksikin tahoksi. Tämän jälkeen järjestelmä osaa tarkistaa, onko tunnistautuneella taholla riittävät käyttöoikeudet järjestelmän käyttämiseen, eli se autorisoi käyttäjän tekemiä toimia. Tunnistamismenetelmät voi käytännössä jakaa kahteen tapaan: joko järjestelmä hoitaa itse käyttäjän tunnistamisen ja käyttöoikeuksien tarkistuksen tai sitten se luottaa johonkin toiseen palveluun ja ulkoistaa tunnistamisen sen vastuulle. Kertakirjautumisjärjestelmä on yksi tällainen järjestelmä, joka keskittää monen palvelun kirjautumiset yhden järjestelmän alle. Näin käyttäjän tarvitsee kirjautua tai rekisteröityä vain kerran, ja muut verkkopalvelut voivat kysyä kertakirjautumisjärjestelmältä käyttäjän tiedot ja luottaa siihen.

Kanadalainen Guelphin yliopisto listaa verkkosivuillaan (SSO Benefits 2016) kertakirjautumisen eduiksi käyttäjäystävällisyyden, tietoturvan sekä kustannus- ja aikasäästöt. On käyttäjän näkökulmasta merkittävästi helpompaa muistaa yksi käyttäjätunnus, kirjautua yhden kerran ja saada heti tarvittavat käyttöoikeudet tarvittaviin järjestelmiin. Järjestelmien ylläpitäjien näkökulmasta se lisää myös tietoturvaa, sillä tunnistautuminen on toteutettu vain yhdessä paikassa eivätkä lukuisat paikalliset tunnistautumistoteutukset jää vanhentumaan ja keräämään ongelmia. Tämä luonnollisesti tekee kertakirjautumisjärjestelmästä kriittisen tietojärjestelmän: jos se ei vastaa tai toimi oikein, ei palveluihin voi kirjautua. Palveluiden ylläpitäjät arvostavat säästettyä aikaa, kun ei ole monta paikkaa, jossa käyttäjätunnuksia tulisi hallita.

3 Kertakirjautumisjärjestelmän suunnittelu

3.1 Projektiorganisaatio ja -tavoitteet

G-Works Oy on vuonna 2010 perustettu noin kymmenen henkilöä työllistävä IT-alan yritys. Se keskittyy verkkopalveluihin ja verkkosivustoihin ja tarjoaa yrityksille yhdestä paikasta kaiken, mitä tarvitaan modernin verkkopalvelun tarjoamiseen alusta loppuun. Tämä sisältää palvelukonseptoinnin, projektihallinnan, palvelinylläpidon, teknisen toteutuksen ja visuaalisen suunnittelun.

G-Works Oy:n asiakasorganisaatiossa oli kehitetty aikojen kuluessa useita verkkopalveluita, muun muassa uutissivustoja, blogeja ja muita sisäisiä järjestelmiä. Näiden järjestelmien käyttöoikeuksien hallinta oli kömpelöä, sillä niitä jokaista täytyi hallita niiden omista hallintapaneeleista. Tämä jätti suuren riskin, että joitain käyttöoikeuksia ei ymmärretty tai osattu antaa organisaation työntekijöille ja että joitain käyttöoikeuksia pitäisi saada poistettua tehokkaasti tietoturvasyistä.

Ongelman ratkaisemiseksi G-Works Oy:ltä tilattiin tekninen ratkaisu, jota lähti kehittämään asiantuntijatiimi. Tiimi koostui asiakasorganisaation edustajasta ja projektipäälliköstä, joiden vastuulla olivat määrittely, projektinhallinta ja hyväksyntätehtävät, graafikosta, jonka vastuulla oli suunnitella järjestelmän visuaalinen ilme ja käyttöliittymät, ja omana tehtävänäni oli toimia tiimin kehittäjänä ja teknisenä suunnittelijana, minkä tein insinöörityönäni. Projektin alussa hankittiin ulkoista konsulttiapua teknistä määrittelytyötä varten.

Tarvekartoituksen pohjalta tekniseksi ratkaisuksi muodostui keskitetty käyttäjätietokanta, joka laajennettiin kertakirjautumisjärjestelmäksi tarjoamaan parempaa käytettävyyttä loppukäyttäjälle. Kertakirjautumisjärjestelmä on tarkoitettu ensisijaisesti organisaation henkilöstön käyttöön, tarjoamaan kaikki palvelut yksillä tunnuksilla ja yhdellä kirjautumisella. Tällä helpotetaan käyttäjähallintaa keskittämällä kaikkien käyttöoikeusmäärittelyt yhden järjestelmän alle. Mahdollisten käyttäjätunnuksiin tai käyttöoikeuksiin liittyvien epäselvyyksien selvittäminen on yksinkertaisempaa, kun on vähemmän paikkoja joissa ne voi olla ristiriidassa toistensa kanssa. Tällä saavutetaan myös merkittäviä tietoturvahyötyjä, sillä vanhentuneita käyttöoikeuksia ei jää päälle eikä voimaan yhtä todennäköisesti, kuin ne jäisivät jossakin vähäisessä käytössä olevassa verkkopalvelussa. Samoin erilaisiin käyttöoikeusmuutoksiin on helpompi reagoida ja käyttäjien toimia helpompi seurata, kun kirjautumiset kiertävät yhden järjestelmän kautta.

Luomalla kaikille organisaation henkilöstössä tunnukset kertakirjautumisjärjestelmään pyritään parantamaan verkkopalveluiden käyttöönottoa ja yhteistyötä. Aikaisemmin organisaation verkkopalvelut olivat jääneet vähäiselle käytölle niiden käyttöönoton hankaluuden vuoksi. Kun käyttäjätunnukset ovat kaikilla jo tiedossa ja käyttöoikeudet annettu, on todennäköisempää, että sekä olemassa olevia että tulevia järjestelmiä kokeillaan ja käytetään herkemmin.

3.2 Järjestelmän vaatimusmäärittely

Kertakirjautumisjärjestelmää tulee voida kehittää modulaarisesti siten, että siihen voidaan kehittää uusia toiminnallisuuksia ilman, että se vaatii jo kehitettyjen toiminnallisuuksien muokkaamista. Toisin sanoen, se ei saa vaikuttaa toiminnassa oleviin järjestelmiin. Käytännössä tämä saavutaan luomalla selkeät rajapinnat, joiden avulla kertakirjautumisjärjestelmä keskustelee muiden järjestelmien kanssa.

Kertakirjautumisjärjestelmän piiriin tulee voida liittää määrittämättömiä asiakassovelluksia. Niitä hallitaan kertakirjautumisjärjestelmän hallintapaneelista vain tietyillä käyttöoikeuksilla. Asiakassovellusten tulee hyödyntää kertakirjautumisjärjestelmää siihen luotavan standardin ja rajapinnan välityksellä, ja näiden asiakassovellusten tulee olla tunnistettavissa luotettavasti, ettei sovellusta voida liittää luvattomasti kertakirjautumisjärjestelmään. Asiakassovellukseen liitetään sallittuja ryhmiä, ja kertakirjautumisjärjestelmä välittää nämä ryhmät asiakassovellukselle, mikäli käyttäjä on jossakin näistä ryhmistä. Kuvassa 2 havainnollistetaan kertakirjautumisjärjestelmän toimintaperiaate.

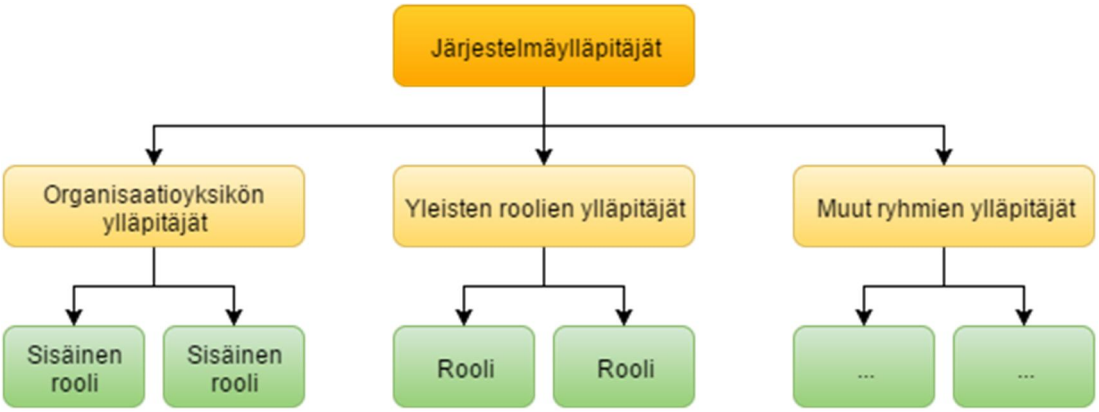
Kuva 2. Kertakirjautumisjärjestelmän toimintaperiaate yksinkertaistettuna.

Rajapinnan täytyy osata erottaa sallitut ja halutut viestit virheellisistä ja väärennetyistä viesteistä. On tärkeää, että käyttäjistä ei välitetä tietoja sellaisille tahoille, joilla ei ole oikeuksia niiden käsittelyyn. Samoin rajapinnan täytyy hylätä vanhentuneet viestit, jottei käyttäjä vahingossakaan jättäisi muiden painettavaksi mitään klikattavaa linkkiä, jolla voisi kirjautua toisen tunnuksilla.

Kertakirjautumisjärjestelmän tulee hallinnoida käyttöoikeuksia hierarkkisesti ryhmittäin. Kaikki henkilöt, jotka kuuluvat johonkin ylläpitoryhmään, voivat hallita sen alapuolella oleviin ryhmiin kuuluvien henkilöiden käyttöoikeuksia.

Kuten kuvasta 3 nähdään, ylin käyttöoikeusryhmä on kertakirjautumisjärjestelmän hallitsijat, jotka voivat hallita kaikkia järjestelmän asetuksia, liittää uusia sovelluksia kertakirjautumisjärjestelmän piiriin ja hallita kaikkien muiden käyttöoikeuksia. Tämän alapuolella

olevat ryhmät ovat ylläpitoryhmiä, jotka on luokiteltu organisaation rakenteen mukaan, yleisten roolien mukaan tai muuten halutun tunnisteen mukaan. Näistä esimerkki voisi olla organisaation ulkoisille konsulteille tarkoitettu ryhmä. Yksi käyttäjä voi olla asetettu moneen ryhmään, niin ylläpitoryhmiin kuin käyttäjäryhmiin.



Kuva 3. Järjestelmän käyttöoikeushierarkia: ylemmällä tasolla olevat hallitsevat alemmalla tasolla olevia.

Kertakirjautumisjärjestelmän täytyy skaalautua suurille käyttäjämäärille, ja sillä täytyy pystyä hallitsemaan useita käyttäjiä samanaikaisesti, jottei sen käytöstä tule työlästä. Tämä tarkoittaa, että käyttäjiä täytyy pystyä hakemaan heidän tietojensa perusteella ja yksittäisten ryhmien kaikki käyttäjät täytyy pystyä siirtämään ja liittämään ryhmistä toisiin.

Käyttäjien täytyy pystyä näkemään omat tietonsa kirjautumalla suoraan kertakirjautumisjärjestelmän hallintapaneeliin, muokkaamaan olennaisimpia tietoja ja suorittamaan tavanomaiset kirjautumiseen liittyvät toimet. Tällaisia toimia ovat kirjautuminen, uloskirjautuminen, salasanan vaihto ja palautus sekä rekisteröityminen. Järjestelmäylläpitäjien tulee pystyä seuraamaan käyttäjien kirjautumisia, jotta nähdään kertakirjautumisjärjestelmän käyttötapoja ja se, onko sen pohjalta syytä tehdä muutoksia. Heidän täytyy pystyä hallitsemaan ja seuraamaan kertakirjautumisjärjestelmän asetuksia ja sähköpostiviestejä.

Asiakassovelluksien täytyy pystyä kysymään kertakirjautumisjärjestelmältä käyttäjistä tietoja niiltä osin, kuin se olisi tavallisestikin saatavilla niille. Käytännössä tämä tarkoittaa käyttäjätilien aktiivisuustiloja ja yhteystietoja.

3.3 Kertakirjautumisjärjestelmän kehitysperusteet

Itse tehdylle verkkopalvelupohjaiselle kertakirjautumisjärjestelmälle on muitakin vaihtoehtoja, kuten Facebookin tai Googlen tarjoamat valmiit kertakirjautumispalvelut. Nämä ja muut ulkoiset tarjoajat kuitenkin rajattiin pois, sillä niiden laajentaminen käyttöoikeuksien hallintaan olisi ollut haastavampaa. Samoin silloin kaikki kirjautumis- ja käyttäjädata olisi ollut ulkoisen tahon omistuksessa, mikä koettiin kriittiseksi puutteeksi, ja samalla kaikki organisaation palvelut olisivat olleet riippuvaisia ulkoisesta toimijasta.

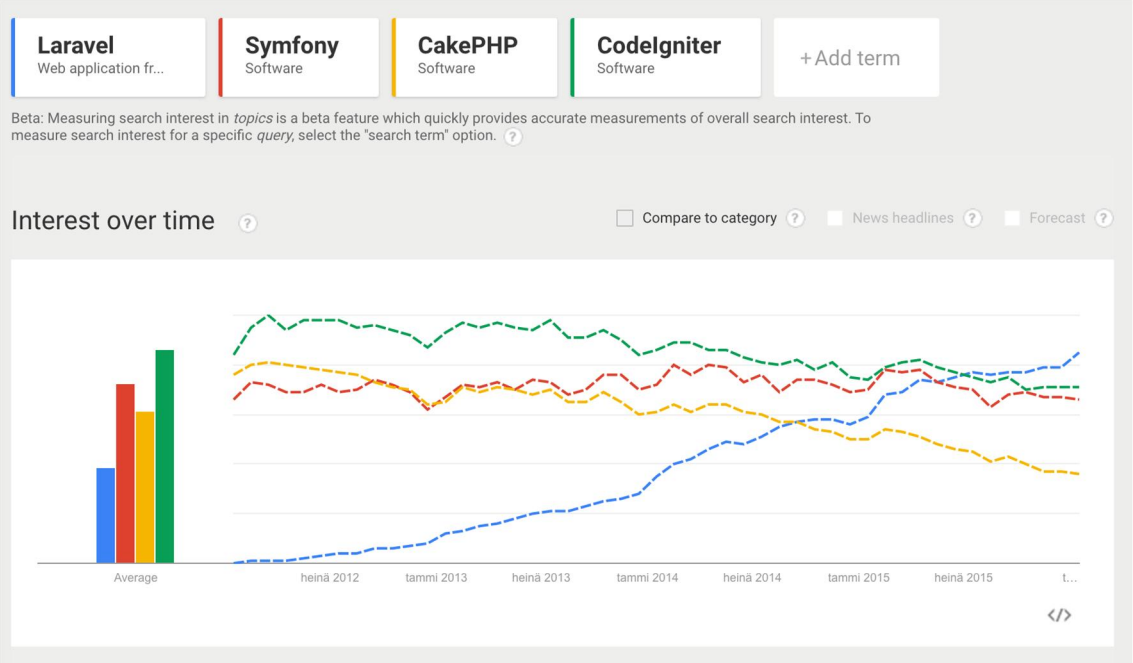
Toteuttamalla räätälöity järjestelmä saadaan parhaat mahdollisuudet vaikuttaa siihen, miten ja missä kertakirjautumisjärjestelmä otetaan käyttöön ja kuinka sitä käytetään käyttöoikeuksien hallintaan. Näin ei myöskään rajoiteta tulevaisuuden mahdollisuuksia laajentaa järjestelmän käyttötarkoitusta tai toiminnallisuuksia. Ulkoisissa palveluntarjoajissa on riskinä, että ne kehittävät palveluaan epäedulliseen suuntaan sitä käyttävän tahon näkökulmasta. Vaihtoehtoisesti palveluiden käyttöehdot saattavat muuttua, jolloin sen käyttö ja jatkokehitys saattaa olla mahdotonta.

4 Järjestelmän teknologiaratkaisut

4.1 Ohjelmistokehys

G-Works Oy on tehnyt paljon verkkosivustoja PHP-kielellä, ja se oli luontevin kieli lähteä toteuttamaan kertakirjautumisjärjestelmää. Pohdittaessa, tehdäänkö kaikki alusta vai hyödynnetäänkö jotain ohjelmistokehystä ohjelmointityön tukena, päädyttiin siihen, että jokin ohjelmistokehys on hyvä ottaa käyttöön. Näin lähdekoodin rakenne pysyy yhtenäisenä eikä kaikkia tiedonkäsittelyyn liittyviä toiminnallisuuksia tarvitse kehittää itse.

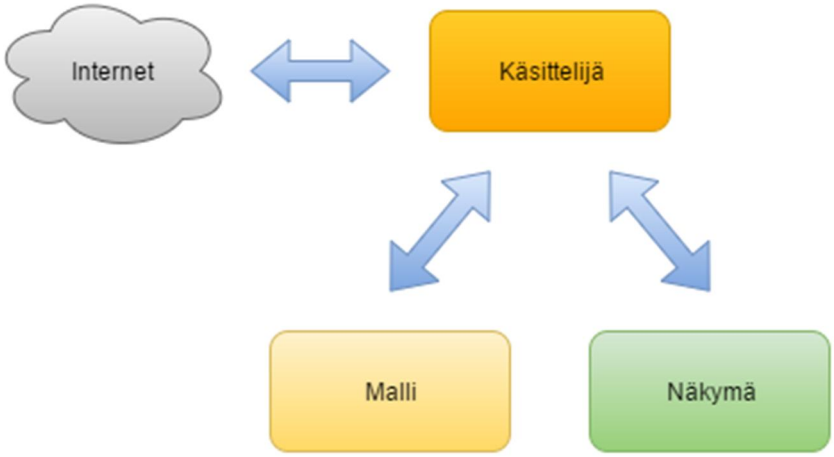
Ohjelmistokehykseksi valittiin Laravel-niminen ohjelmistokehys, sillä siinä oli hyvä dokumentaatio, hyvä noste kehitysyhteisöissä ja paljon opetusmateriaalia saatavilla. Kuvassa 4 on verrattu eri PHP-ohjelmistokehyksien Google-hakuja, joissa kiinnostus Laraveliin on ollut selkeän nousujohteinen. Sen tarjoamat työkalut toimisivat hyvin järjestelmän kehityksessä. Järjestelmää kehitettäessä Laravelin viimeisin julkaistu versio oli versio 3.



Kuva 4. Eri PHP-ohjelmistokehyksien hakutiheys Googlesta Google Trendin mukaan (Google Trends).

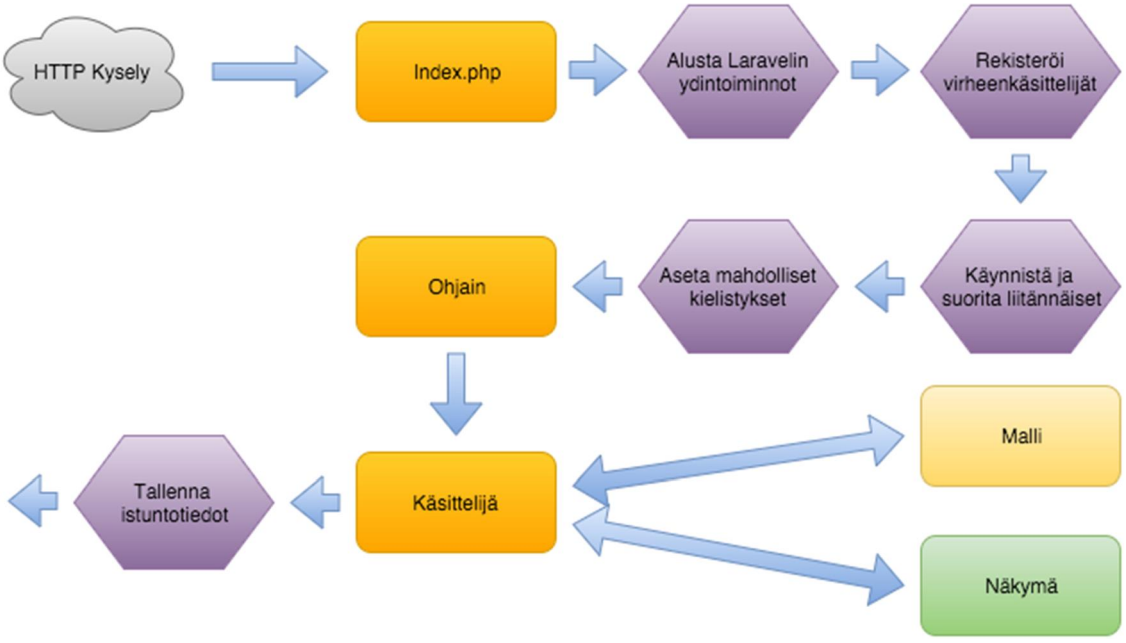
Laravel 3 on niin sanottu Model View Controller (MVC) -mallin ohjelmistokehys (Rees 2012: 10). Tämä tarkoittaa, että ohjelmakoodi on jaoteltu kolmeen tärkeään rooliin: malliin (Model), näkymään (View) ja käsittelijään (Controller). Mallin vastuulla on toimia järjestelmän tiedon tallentajana ja hallitsijana. Näkymän vastuulla on huolehtia siitä, että järjestelmä näyttää oikeat asiat oikeassa paikassa oikean näköisenä ja toimii esimerkiksi käyttöliittymänä. Käsittelijän vastuulla on hallita näkymän ja mallin tilaa riippuen sille tulleista pyynnöistä, esimerkiksi käyttäjän syötteistä.

Kuva 5 kuvaa MVC-mallin toimintaa perinteisen verkkosivukyselyn osalta. Sivusto ladataan verkko-osoitteella, jolloin käsittelijä saa pyynnön, hakee mallilta tarvittavat tiedot ja lähettää ne näkymälle sivuston muodostamista varten ja näyttää sen.



Kuva 5. Laravel-ohjelmistokehyksen malli-näkymä-käsittelijäperiaate ja tiedonkulku.

Laravel ohjaa tulevan verkkokyselyn ensin public-hakemiston index.php-tiedostoon, joka alustaa varsinaisen ohjelmistokehyksen ja koko järjestelmän. Kysely seuraa kuvassa 6 esitettyä kaavaa, josta havaitaan myös, kuinka MVC-malli istuu Laravelin kokonaiskuvaan.



Kuva 6. Kyselyn eteneminen Laravel-ohjelmistokehyksessä MVC-mallin mukaisesti (tehty pohjautuen Laravel 3 -lähdekoodiin).

Laravel tekee paljon kyselyn taustalla, mikä antaa ohjelmoijan keskittyä liiketoimintakriittisen ohjelmakoodin tuottamiseen eikä taustajärjestelmien ohjelmointiin. Kuvan 6 vaiheista vain ohjaimen määrittely ja käsittelijöiden, mallien ja näkymien luonti on ohjelmoijan vastuulla, muut toiminnallisuudet Laravel tarjoaa taustalla.

4.2 Kyselyiden reititys

Laravelissa kyselyjä ohjataan oikeisiin toimintoihin reitittämällä ne ohjaimessa määriteltyjen sääntöjen mukaan halutuille käsittelijöille. Tähän on olemassa valmis ”Router”-reititinluokka, jonka tehtävänä on reitittää tietty verkkokysely tiettyyn toimintoon. Sillä voidaan samalla määrittää kyselyyn haluttuja suodattimia ja tarkistuksia, jotka suoritetaan, ennen kuin kysely päätyy haluttuun toiminnallisuuteen.

Kuvassa 7 on osa kertakirjautumisjärjestelmän asiakassovelluksien hallintaan käytetyistä reitityksistä. Siinä reitit on ryhmitetty omaan ryhmään, jolle on määritetty, että kaikkien siihen kuuluvien reittien täytyy salata niiden liikenne ja ennen varsinaista toiminnallisuutta suoritetaan ”auth”- ja ”role”-nimiset suodattimet. Nämä suodattimet tarkistavat, että kyselyn suorittaja on autentikoitunut ja että sillä on järjestelmäylläpitäjän oikeudet.

```
Route::group(array(
  'https' => true,
  'before' => 'auth|role:administrate'), function () {

  Route::get('applications', array(
    'as'    => 'applications',
    'uses'  => 'applications@index'
  ));
  Route::get('applications/(:any)', array(
    'as'    => 'application',
    'uses'  => 'applications@show'
  ));
  Route::get('applications/new', array(
    'as'    => 'new_application',
    'uses'  => 'applications@new'
  ));
  Route::get('applications/(:any)/edit', array(
    'as'    => 'edit_application',
    'uses'  => 'applications@edit'
  ));
  Route::get('applications/(:any)/groups', array(
    'as'    => 'new_application_group',
    'uses'  => 'applications@index'
  ));
  Route::get('api/applications', array(
    'as'    => 'api_applications',
    'uses'  => 'applications@index'
  ));
});
```

Kuva 7. Ohjain, jossa on määritetty asiakassovelluksien hallintaan liittyvät reitit.

Yksittäiset ”Route::get”-komennot määrittävät, minkälaista verkkokyselyä reitti vastaa. Näissä tapauksissa tavanomaisia GET-kyselyjä ohjataan ”applications”-nimiselle ohjaimelle tiettyihin toimintoihin. Esimerkiksi kun tehdään GET-kysely osoitteeseen ”/applications”, se ohjataan käyttöoikeustarkistuksien jälkeen ”uses”-kohdan mukaisesti ”application”-käsittelijän ”index”-toimintoon. Tämä ”application”-käsittelijä vastaa asiakassovelluksien toiminnoista, ja ”index”-toiminto näyttää asiakassovelluksien listanäkymän.

Reitit voidaan nimetä ”as”-kohdalla, esimerkiksi listanäkymä on nimetty ”applications”-nimiseksi. Tämän avulla voidaan myöhemmin tehdä uudelleenohjauksia tai linkkejä suoraan reitityksen nimellä, jolloin reitityksen osoitteen muuttuessa sitä ei tarvitse päivittää muualle.

Reitysten osoitteet tukevat muuttuvaa ja yksilöivää tietoa, kuten edellä esitetyssä esimerkissä nähdään ”edit_application”-nimisestä reitistä, joka on asiakassovelluksen muokkaamiseen tarkoitettu näkymä. Siinä osoite on muotoa ”applications/(:any)/edit”, ja tämä tarkoittaa, että reitti vastaa kaikkia osoitteita, joissa ”applications”- ja ”edit”-kohdat ovat kiinteitä ja vain ”(:any)”-kohta muuttuu. Tässä tapauksessa siinä haetaan sovelluksen tunnistetta, jonka reititin osaa välittää ohjaimelle, jotta se osaa näyttää muokkausnäkymässä oikean asiakassovelluksen tiedot.

Reititin tukee kaikkia muitakin HTTP-protokollan (Hyper Text Transfer Protocol) mukaisia kyselyjä, kuten POST-, PUT- ja DELETE-kyselyjä. Ohjelmistokehys tunnistaa tehdyn kyselyn HTTP-tunnisteista tai niiden mukana lähetetystä ”_input”-nimisestä kentästä.

4.3 Tietokanta

Kertakirjautumisjärjestelmän tietomallit ovat vahvasti relaatiopohjaisia. Järjestelmässä täytyy tallentaa käyttäjätunnuksia, joihin liittyy käyttäjäryhmiä, joihin liittyy taas hierarkkisia ryhmiä ja useita asiakassovelluksia. Samoin on olennaista, että käyttäjän tekemät toimet, kuten käyttöoikeusmuutokset tai kirjautumishistoria, voidaan todentaa. Käyttäjiin liittyy siis paljon tietoa, jota on luontevaa kuvata relaatioilla.

Eri relaatiotietokannoista valittiin MySQL-tietokanta, sillä se oli entuudestaan varsin tuttu, ja käytetyt työkalut osasivat hyödyntää MySQL-tietokantoja ilman lisätyötä. Tietokannalta vaadittiin tietoturvaa, joten siinä tehtiin kertakirjautumisjärjestelmälle kokonaan oma tietokantainstanssi, jotta palvelimen muut mahdolliset tietokannat eivät tietäisi kertakirjautumisjärjestelmästä yhtään mitään. Järjestelmän normaalikäyttöön ja ylläpitotoimiin luotiin omat tietokantatunnukset kerroksittaisen tietoturvan mukaisesti. Näin ollen, vaikka joku pääsisikin tekemään tietokantaan toimia, se ei saisi täyttä pääsyä ja esimerkiksi tuhottua sitä kokonaan.

Salasanojen kanssa haluttiin olla erityisen varovaisia ja käyttöön otettiin mahdollisimman vahva salaus: bcrypt-salaus vaikeuskertoimella 10. Näin ollen, vaikka joku saisi koko tietokannan käsiinsä, hän ei näkisi suoraan käyttäjien salasanoja selkokielisinä, vaan ne olisivat salattuina. Bcryptin varmentaminen on hidas operaatio: riippuen vaikeuskertoimesta, se voi kestää esimerkiksi puoli sekuntia, mikä ei ole yksittäisen käyttäjän salasanan varmennukseen liian kauan. Koneellisesti oikean salasanavariaation löytäminen

kaikkien mahdollisuuksien seasta puolen sekunnin iteraatioilla on liian hidasta, mikä tekee bcryptistä tietoturvallisen (Franco 2014: 99).

Tietokantarakenteen suunnittelussa täytyi ottaa huomioon, että sitä voidaan laajentaa myös muihin käyttötarkoituksiin, esimerkiksi organisaation ulkoisten käyttäjien tietojen varastointiin.

Koska järjestelmä ja sen tietokanta rakennettiin laajeneminen mielessä, tietokannan tuli tukea useaa salasanaa ja kirjautumistapaa yhdelle käyttäjille. Tämän vuoksi tehtiin valtavirrasta poikkeavasti oma taulu salasanoille, jotka oli yksilöity käyttäjälle ja kirjautumistavalle. Tämä mahdollisti sen, että käyttäjä voisi teoriassa jatkossa kirjautua tavallisesti kertakirjautumisjärjestelmän kirjautumisdialogista tai muun ulkoisen toimijan tarjoamasta tunnistautumispalvelusta, jolloin kertakirjautumisjärjestelmä toisi siihen käyttöoikeushallinnan.

4.4 Toteutustavat

Kertakirjautumisjärjestelmän kehityksessä tultiin nopeasti tilanteeseen, jossa oli yksi versio tuotannossa, yksi asiakkaan hyväksynnässä ja yksi kehitteillä. Versioiden eroja selventämään otettiin käyttöön semanttiset versionumerot. Semanttisessa versioinnissa versionumero muodostuu kolmesta osasta: MERKITTÄVÄ.VÄHÄINEN.PÄIVITYS, eli esimerkiksi 1.9.2. Merkittävä versionumero kasvaa, kun tehdään päivityksiä, jotka rikkovat taaksepäinyhteensopivuuden. Vähäinen versionumero muuttuu, kun tuodaan lisää ominaisuuksia, mutta ei rikota taaksepäinyhteensopivuutta. Päivitysversionumero muuttuu, kun tehdään korjauspäivityksiä. (Semantic Versioning 2010).

Kertakirjautumisjärjestelmä toteutettiin iteroiden, mutta kuitenkin aina projektiluonteisesti. Yksi projekti koostui suunnittelusta, graafisesta suunnittelusta, teknisestä toteutuksesta ja julkaisusta. Julkaisun jälkeen sopivaa versionumeroa nostettiin. Graafisessa suunnittelussa työt kävivät useita kierroksia asiakkaalla kommentoitavana, minkä jälkeen ne hyväksyttiin. Tämän jälkeen teknisessä toteutuksessa järjestelmää kehitettiin noin viikon tai kahden viikon kestoisissa sykleissä, minkä jälkeen sitä näytettiin asiakkaalle ja saatiin palautetta, jonka pohjalta sitä voitiin ohjata eteenpäin.

Kehitystyötä johdettiin käytännössä kahdella eri tavalla. Iteraatioiden varsinaista kehitystyötä seurattiin kanbanista ja scrumista johdetulla tavalla, jossa kehityksen eri tehtäviä kuvasivat värikkäät muistilaput valkotaululla. Niitä siirrettiin eri vaiheita kuvastavista sarakkeista toisiin, ja näin kehitysvaiheen tilanne oli helposti nähtävillä yhdellä silmäyksellä.

Kun versio siirtyi pois varsinaisesta kehitysvaiheesta muihin ympäristöihin ja ylläpitoon, tehtävien hallinta siirtyi tikettipohjaiseksi. Tiketit ja niihin liittyvät korjaukset pystyttiin helposti sitomaan johonkin tiettyyn versioon. Tikettijärjestelmänä käytettiin Git-pohjaista versiohallintajärjestelmää nimeltä GitLab, johon on liitetty erittäin helppokäyttöinen tiketinseurantatoiminnallisuus. Tämä helpotti työn ja lähdekoodin seuraamista, kun ei ollut useaa eri järjestelmää, joita seurata.

Versiohallintastrategiana hyödynnettiin Gitin mahdollistamia versiohaaroja. Yksi "master"-niminen haara kuvasti järjestelmän senhetkistä tuotantoversiota. Sen lisäksi käytettiin versiohaaroja, jotka olivat nimetty "dev-XYZ"-tyyppisesti, jossa XYZ oli kehitettävä versionumero. Näistä versiohaaroista oli aina jokin esillä hyväksyntätestausympäristössä. Master- tai Dev-haaroihin ei saanut tehdä suoraan muutoksia, vaan kehitystä varten tehtiin omia toiminnallisuushaaroja, jotka nimettiin kehitettävän toiminnallisuuden mukaan. Näitä toiminnallisuushaaroja ei lähetetty GitLabiin, vaan sinne lähetettiin vain Master- ja Dev-haarat, joihin toiminnallisuushaarat yhdistettiin.

4.5 Palvelinympäristöt ja ylläpito

Projektissa haluttiin, että varsinainen kehitystyö tehtäisiin asiakasorganisaation omistamalla palvelimella ja kaikki projektiin liittyvä ohjelmakoodi olisi organisaation omistuksessa. Palvelinta tai ympäristöä ei kuitenkaan ollut entuudestaan olemassa, vaan sellainen hankittiin ja rakennettiin kehitystyön ohella. Asiakasorganisaatiolle hankittiin kokonaan oma fyysinen palvelin kolmannen osapuolen laitesaliin pilvipalveluiden sijasta. Tällä vähennettiin riippuvuuksia ulkoisiin palveluntarjoajiin ja saatiin palvelimet Suomeen.

Suomalaisuus oli G-Worksille ja organisaatiolle tärkeää, sillä Edward Snowdenin tuotua Yhdysvaltain kansallisen turvallisuusviraston (National Security Agenct, NSA) PRISM-

vakoiluohjelman (Planning Tool for Resource Integration, Synchronization, and Management) julkisuuteen (Gellman & Poitras 2013) G-Works ja asiakasorganiosaatio halusivat markkinoida itseään suomalaisena yrityksenä.

Luotuun ja rakennettuun ympäristöön tuotiin myöhemmin kaikki asiakasorganisaation muut verkkopalvelut. Samoin haluttiin mahdollistaa päällekkäinen kehitys ja julkaisu luomalla kehitysympäristö, hyväksyntätestausympäristö ja tuotantoympäristö. Järjestelmän ja palvelinympäristön haluttiin kestävän alkuvaiheessa 200 käyttäjää ja niiden hallinnointia, mutta skaalautuvan aina 10 000 käyttäjään asti. Palvelinympäristön osalta tähän varauduttiin ottamalla riittävän tehokas palvelin, jossa olisi laajennusmahdollisuuksia muistin, prosessorin ja levytilan osalta.

Projektissa käytettiin hyväksi havaittua ympäristömallia, jossa kehitetään joko keskitetyllä kehityspalvelimella tai paikallisessa kehitysympäristössä. Kehityksestä versio siirtyi hyväksyntätestausympäristöön, jossa sekä G-Works että tilaaja pääsivät näkemään ja testaamaan seuraavaksi julkaistavaa versiota, tekemään sisällönsyöttöä ja lopuksi hyväksymään projektin. Tästä se siirtyi tuotantoon loppukäyttäjien käyttöön ja testattavaksi.

Erillisillä ympäristöillä mahdollistettiin jouheva kehitys, jossa ei tarvinnut pysäyttää kehitystä sillä välin, kun sidosryhmillä oli tarpeita tehdylle järjestelmälle. Tällaisia tarpeita ovat esimerkiksi asiakkaan tekemä hyväksyntätestaus ja järjestelmän esittelytilaisuudet, joissa palvelu ei saa mennä rikki. Erilliset ympäristöt olivat kriittisiä, jotta pystyttiin ylipäätänsä testaamaan uusia toiminnallisuuksia, ennen kuin ne päätyivät asiakkaan tai loppukäyttäjän nähtäväksi.

Teknisesti keskitetty kehitysympäristö ja hyväksyntätestausympäristö olivat samalla palvelimella mutta eri hakemistoissa ja eri verkko-osoitteiden alla. Tuotantoympäristö oli oma palvelimensa tietoturvan ja toiminnan varmistamiseksi. Julkaisut paikallisesta kehitysympäristöstä keskitettyyn kehitysympäristöön tai hyväksyntätestausympäristöön tehtiin versiohallintaa ja GitLabia hyväksikäyttäen. Kun oli saatu jokin toiminnallisuus valmiiksi, se lähetettiin GitLabiin, josta se käytiin lataamassa eri ympäristöihin. Kaikki nämä oli tehty mahdollisimman helposti ohjelmallisesti, jolloin siirrot olivat vain muutama komento komentorivillä.

5 Järjestelmän keskeisimmät toiminnallisuudet

5.1 Sisäinen tunnistautuminen

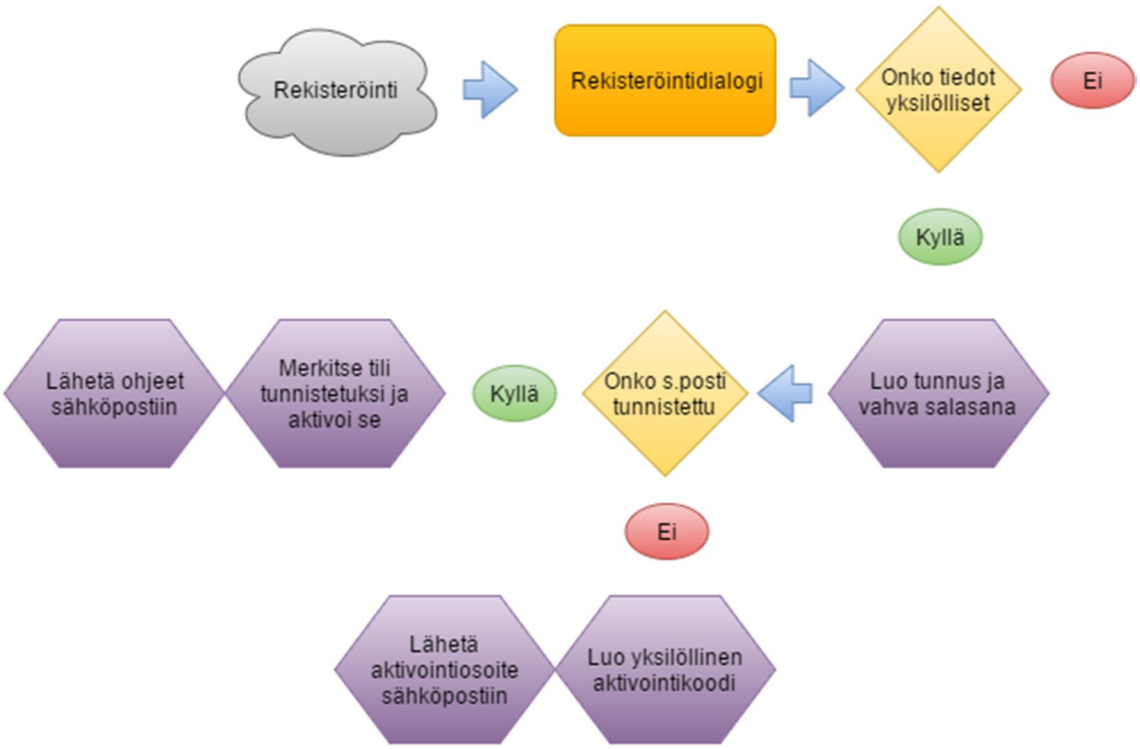
Rekisteröinti

Melko pian sen jälkeen, kun kertakirjautumisjärjestelmä oli otettu organisaation sisäiseen käyttöön, se laajennettiin myös organisaation ulkopuolisille henkilöille. Käytännössä he olivat organisaation verkkopalveluiden käyttäjiä, jotka käyttivät kertakirjautumisjärjestelmää tunnistettuun kommentointiin. Tämä tarkoitti, että järjestelmään täytyi pystyä rekisteröitymään itsenäisesti, ilman sidoksia organisaatioon tai ilman siltä vaadittuja toimenpiteitä.

Haluttiin kuitenkin erottaa organisaation sisällä tunnistetut tai tunnetut henkilöt ulkopuolisista ja tuntemattomista henkilöistä. Näin organisaation tunnistamia henkilöjä voidaan korostaa esimerkiksi keskustelualueilla ja robottien luomat käyttäjätunnukset poistaa.

Rekisteröitymisestä tehtiin kaksivaiheinen, jossa varmennetaan käyttäjän sähköpostiosoite oikeaksi ja aktiiviseksi sekä annetaan käyttäjälle oletuksena vahva salasana. Näin voidaan varmistua, että käyttäjätunnukseen on yhdistetty oikea sähköpostiosoite.

Kuvassa 8 näkyy, mitä kertakirjautumisjärjestelmä tekee rekisteröitymisen yhteydessä. Käyttäjän näkökulmasta rekisteröitymisprosessi on varsin tuttu: syötä sähköposti ja saat aktivointilinkin ja salasanan sähköpostiisi. Taustalla kuitenkin kertakirjautumisjärjestelmä tarkistaa, että kaikki tiedot ovat yksilölliset ja että syötetty sähköpostiosoite on esitunnistettujen listalla.



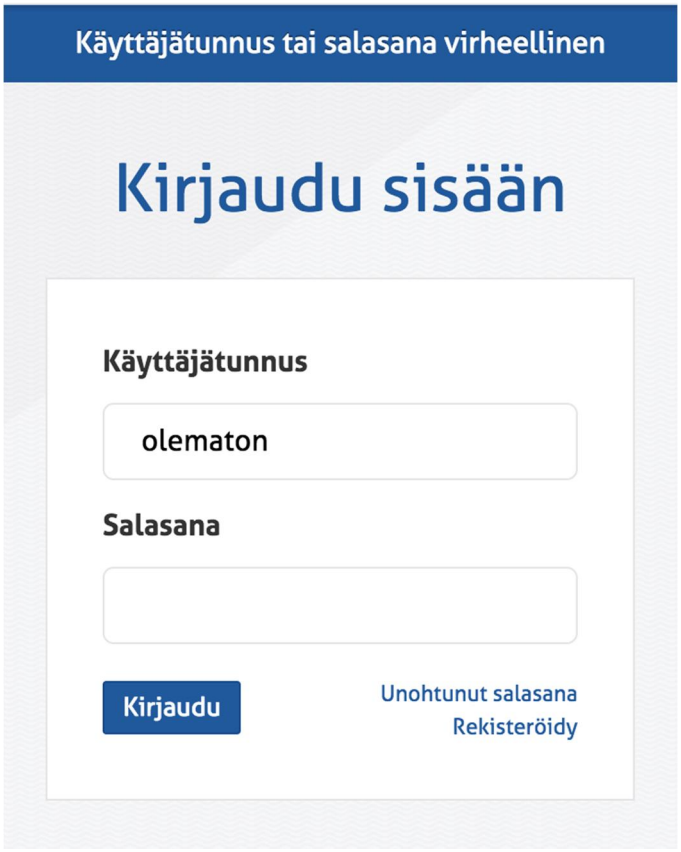
Kuva 8. Kertakirjautumisjärjestelmän tekemät toimet rekisteröitymisen yhteydessä.

Esitunnistettujen käyttäjien ei tarvitse aktivoida tiliään erikseen minkään määräajan sisällä, vaan tili on suoraan käytössä ja sen käyttö opastetaan automaattisella sähköpostiviestillä. Esitunnistettujen sähköpostiosoitteiden käyttö toteutettiin valmiiksi luotujen tunnusten sijaan, jotta käyttöönottoprosessi olisi aktiivinen käyttäjän toimesta. Käyttäjätunnuksien ei haluttu unohtuvan sähköpostien joukkoon ja käyttöönoton unohtuvan. Samalla saatiin seurattua rekisteröitymistiheyttä paremmin.

Käyttäjätunnukset aktivoidaan yksilöllisellä aktivointilinkillä, joka tulee rekisteröitymisen jälkeen sähköpostiin. Tätä linkkiä painamalla päätyy kertakirjautumisjärjestelmän hallintapaneeliin, omaan profiilinäkymään. Tästä näkymästä käyttäjä voi vaihtaa oman salasanan tai siirtyä käyttämään asiakassovelluksia, joihin hänen ryhmänsä oikeuttavat. Aktivointilinkki itsessään on voimassa vain 15 minuutin ajan, minkä jälkeen aktivointilinkkiä painamalla päätyy uudestaan rekisteröitymään. Kertakirjautumisjärjestelmän palvelimella suoritetaan minuutin välein sovellus, joka poistaa käyttäjätilit, joita ei ole aktivoitu, ja joilla ei ole voimassaolevaa aktivointilinkkiä.

Kirjautuminen

Kun käyttäjä on rekisteröitynyt tai hänelle on luotu tunnukset järjestelmään ja tili on aktiivinen, hän voi kirjautua kertakirjautumisjärjestelmän hallintaliittymään. Hallintaliittymän toiminnallisuudet riippuvat käyttäjän käyttöoikeuksista, mutta vähimmäistasolla käyttäjä näkee omat tietonsa ja sen, mihin ryhmiin hän kuuluu, ja voi vaihtaa salasanansa. Kuvassa 9 näkyy kertakirjautumisjärjestelmän kirjautumisdialogi, johon syötetään käyttäjätunnus ja salasana, joiden oikeellisuus tarkistetaan ja virheestä esitetään virheilmoitus.



Kuva 9. Kertakirjautumisjärjestelmän kirjautumisdialogi.

Kertakirjautumisjärjestelmän kirjautumisnäkymään ei voi suoraan mennä jollain verkko-osoitteella, vaan käyttäjä ohjataan kirjautumisnäkymään siinä vaiheessa, kun hän yrittää ladata jotain kirjautumista vaativaa näkymää. Tällä yhtenäistetään kertakirjautumisjärjestelmän toteutusta sisäisen tunnistautumsen ja asiakassovelluksien kautta tulevien tunnistautumisien osalta. Yhtenäistäminen toteutettiin kuljettamalla kaikki tunnistautumiset samaan tunnistautumisputkeen, jossa tehdään tarvittavat tarkistukset käyttäjän istuntojen autenttisuudesta, palautusosoitteista ja käyttäjän syöttämistä tunnuksista.

Kun käyttäjä yrittää ladata jonkin näkymän, jossa täytyy olla kirjautuneena, järjestelmä tarkistaa, onko voimassa olevaa istuntoa. Jos ei ole, se lähetetään edellä mainittuun tunnistautumisputkeen, jossa luodaan uusi yksilöity kirjautumisistunto. Istuntoon tallennetaan osoite, jota on yritetty ladata, ja istunnolle annetaan yksilöllinen tunniste. Tämän tunnisteen avulla käyttäjällä voi olla monta samanaikaista kirjautumisdialogia, esimerkiksi useissa selaimen välilehdissä. Kun käyttäjä kirjautuu yhdessä auki olevista välilehdistä, voidaan automaattisesti todeta käyttäjän kirjautuneen kaikissa muissakin välilehdissä ja ohjata ne oikeille sivuille.

Käyttäjätilit tarkistetaan kysymällä tietokannasta, löytyykö annetulla käyttäjätunnuksella voimassa olevaa käyttäjätunnusta. Ylläpitäjillä on mahdollisuus asettaa käyttäjätilejä tilapäiseen käyttökieltoon, mikäli he näkevät sen tarpeelliseksi esimerkiksi väärinkäytösepäilyissä tai muuten selvittäessään kirjautumisia. Lähetettyyn salasanaan lisättiin järjestelmässä käytetty suolaus, ja ne salattiin bcrypt 10:llä. Lopuksi tätä tulosta verrattiin tietokannassa olevaan vastaavasti tallennettuun salasanaan. Mikäli tunnuksissa on virheitä tai tili ei ole voimassa, siitä näytettiin virheilmoitus kuvassa 9 näkyvässä ilmoitusosiossa.

Salasanapalautukset

Mikäli käyttäjä on unohtanut käyttäjätunnuksensa tai salasanansa, hän pystyy tilaamaan salasanapalautuksen kirjautumisnäkymästä. Käytännössä tämä tarkoitti sitä, että käyttäjä syöttää sähköpostiosoitteen, joka on liitettynä hänen käyttäjätiliinsä, tunnusten palautuslomakkeeseen. Sähköposti on tiedettävä ja siihen on oltava pääsy, tai automaattista salasanapalautusta ei voi tehdä. Mikäli näitä tietoja ei ole, käyttäjän täytyy olla yhteydessä järjestelmäylläpitäjiin, jotka voivat nollata salasanan, mikäli he kokevat perustelut riittäviksi.

Kun käyttäjä tilaa salasanapalautuksen, hänelle luodaan yksilöllinen tunniste salasanapalautukseen. Kun tunniste on luotu, käyttäjän sähköpostiin lähetetään viesti, joka sisältää ohjeet ja verkko-osoitteen, josta salasanan voi vaihtaa. Osoitteeseen on sisällytetty juuri luotu yksilöllinen koodi, joka tarkastetaan, ennen kuin salasanan voi nollata, eikä sitä tarvitse erikseen syöttää käsin. Tunniste on voimassa neljä tuntia sen tilauksesta, eikä uutta voi tilata tämän neljän tunnin aikana, jos sellainen on jo voimassa. Tällä pyritään välttämään useita päällekkäisiä salasananollauksia, joista voisi koitua vikatilanteita tai häiriötä käyttäjille.

Salasana on aina muutettava palautuksen yhteydessä, sillä kertakirjautumisjärjestelmällä ei ole yksisuuntaisen salaustekniikan vuoksi kykyä lukea tallennettua salasanaa ja kertoa sitä käyttäjälle. Salasana ei kuitenkaan muutu, ennen kuin se on erikseen vaihdettu sähköpostiin tullutta linkkiä seuraamalla. Ylläpitäjät ja järjestelmäylläpitäjät voivat kuitenkin nollata käyttäjän salasanan käyttäjän hallintapaneelista, mikäli näkevät sen tarpeelliseksi. Käyttäjän vanha salasana poistuu välittömästi käytöstä ja hänelle lähetetään sähköpostitse uusi salasana.

Uloskirjautuminen

Kertakirjautumisjärjestelmästä uloskirjautuminen on suoraviivainen toimi ja perustuu käyttäjän istuntojen puhdistamiseen. Uloskirjautumiselle luotiin hallintaliittymän kaikissa näkymissä oleva painike, joka oli teknisesti uloskirjautumislomake, joka lähetetään järjestelmän käsiteltäväksi, mutta käyttäjälle se näkyi tavallisena painikkeena.

Uloskirjautumispainikkeesta ei tehty tavallista linkkiä, sillä HTTP-spesifikaation mukaan tavallinen linkki ja siitä muodostuva GET-kysely ei saa aiheuttaa sivuvaikutuksia, kuten muutoksia käyttäjän tilaan (IETF 1999). Samalla mahdollistettiin uloskirjautumisen yhteydessä tehtävä tietoturvatarkistus, jolla estetään käyttäjän tahaton uloskirjaaminen. Jos uloskirjautuminen olisi tavallisen linkin takana, joku voisi näyttää käyttäjälle keskustelualueella kuvan, jonka lähdeosoitteena olisi kertakirjautumisjärjestelmän uloskirjautumisosoite. Tämä saisi käyttäjän selaimen lataamaan olematonta kuvaa uloskirjautumisosoitteesta, mikä taas kirjaisi käyttäjän tahattomasti ulos järjestelmästä.

Tietoturvatarkistus, jolla estetään toisen istunnon hyödyntäminen edellä mainitulla tavalla, on niin sanottu Cross Site Request Forgery (CSRF) -tarkistus. Tarkistuksessa varmennetaan, että kyselyt tehdään oikeasta lähteestä, kuten kertakirjautumisjärjestelmän käyttöliittymästä. Varmennus tehdään luomalla käyttäjälle yksilöllinen tunniste, jota päivitetään aina, kun käyttäjän tila on muuttunut, ja lähettämällä se kyselyn mukana ja varmistamalla, että lähetetty tunniste vastaa palvelimen tietämää tunnistetta (Hanging; Zhao 2015: 137–138).

Kun uloskirjautumispyyntö tulee oikeilla tunnisteilla, käyttäjän kaikki kertakirjautumisjärjestelmään liittyvät istuntotiedot yksinkertaisesti pyyhitään ja hänet ohjataan takaisin profiilisivulle. Profiilisivulle ohjaaminen kirjautumattomana taas aloittaa uuden kirjautumisprosessin, ja prosessi alkaa alusta tyhjältä pöydältä.

5.2 Asiakassovellukset

AsiakasSovellusten hallinta

Asiakassovelluksien tunnistamista ja tietoturvaa varten päätettiin hyödyntää julkisen avaimen salaustekniikkaa ja digitaalista allekirjoitusta. Jotta asiakassovellus ja kertakirjautumisjärjestelmä voisivat keskustella toistensa kanssa, niiden täytyy vaihtaa julkisia avaimia keskenään ja allekirjoittaa lähettämänsä viestit omalla salaisella avaimella, jonka voisi tunnistaa ja avata tällä jaetulla julkisella avaimella. Oman julkisen ja salaisen avainparin saa luotua esimerkiksi koodiesimerkissä 1 listatuilla komennoilla asiakassovelluksen Linux-palvelimella.

```
Yksityinen: openssl genrsa -aes256 -out application.priv
2048
Julkinen: openssl rsa -in application.priv -pubout -out ap-
plication.pub
```

Koodiesimerkki 1. Avainparien muodostus Linux-palvelimella.

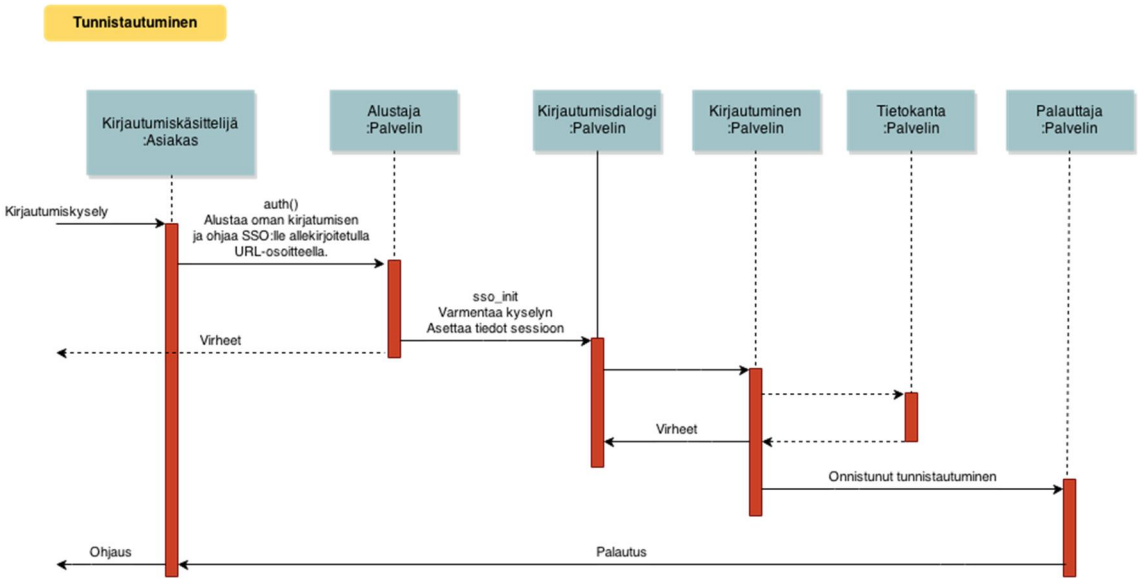
Näin saatu julkinen avain syötetään hallintapaneeliin asiakassovelluksen nimen ja ryhmien kera, ja tämän jälkeen asiakassovellus ja kertakirjautumisjärjestelmä voivat keskustella toistensa kanssa.

Tunnistautuminen asiakassovelluksesta

Asiakassovellukseen kirjaudutaan kertakirjautumisjärjestelmän rajapinnan välityksellä. Asiakassovellus ohjaa käyttäjän kertakirjautumisjärjestelmään ja välittää ohjauksen mukana ennalta määritetyt tunnistetiedot, jotka kertakirjautumisjärjestelmä tarkistaa ja sallii käyttäjän kirjautua sisään. Kirjautumisen jälkeen kertakirjautumisjärjestelmä tallentaa käyttäjän istunnon omalle palvelimelleen, jottei käyttäjän tarvitse kirjautua joka kerta uudestaan, ja ohjaa käyttäjän takaisin asiakassovellukseen ennalta määritettyjen tunnistetietojen kera.

Kuvassa 10 havainnollistetaan yksityiskohtaisella tasolla, kuinka tunnistautumiskysely etenee teknisesti. Käyttäjän lähettämän kirjautumiskyselyn täytyy mennä asiakassovelluksen kirjautumiskäsittelijän läpi, joka muodostaa tarvittavat tunnistetiedot ja ohjaa käyttäjän kertakirjautumisjärjestelmään. Kertakirjautumisjärjestelmä tarkistaa asiakassovelluksen tunnisteet ja alustaa kirjautumisen, jotta se osaa myöhemmin ohjata käyttäjän

takaisin oikeaan paikkaan, vaikka sillä olisi useita kirjautumisikkunoita auki. Tämän jälkeen se näyttää käyttäjälle kirjautumisdialogin tai virheen mahdollisista tunnisteista.



Kuva 10. Tunnistautumiskyselyn eteniminen käyttäjän, asiakassovelluksen ja kertakirjautumisjärjestelmän välillä.

Kirjautumisdialogissa käyttäjä voi syöttää tunnuksensa, jotka järjestelmä välittää kirjautumiskäsittelijälle, joka vahvistaa ne tietokannasta. Mikäli tunnuksissa on jokin virhe, käyttäjälle kerrotaan niistä ja hän voi yrittää uudestaan. Jos kirjautuminen onnistui, järjestelmä ohjaa kyselyn palauttajalle, joka ohjaa käyttäjän takaisin asiakassovelluksen kirjautumiskäsittelijälle oikeiden tunnisteiden ja tietojen kera. Asiakassovellus päättää tämän jälkeen, mitä käyttöoikeuksia käyttäjälle annetaan kertakirjautumisjärjestelmältä saatujen käyttöoikeusryhmien perusteella.

Kertakirjautumisjärjestelmän ja asiakassovelluksien istunnot ovat täysin toisistaan erillisiä. Toisin sanoen, käyttäjä voi olla kirjautuneena vain toiseen järjestelmistä, ilman että se aiheuttaa kummempia ongelmia, sillä oletuksella, että molemmat toimivat oikein. Tästä syystä asiakassovellus käsittelee itse oman uloskirjautumisensa joko ennen kertakirjautumisjärjestelmän uloskirjautumista tai sen jälkeen.

Mahdollinen ongelmakohta voi tulla, jos asiakassovelluksen päässä on tullut virhe kertakirjautumisjärjestelmän käyttöönotossa eikä se osaa kirjata käyttäjää sisään oikein tai sen istuntojenkäsittelyssä on jotain ongelmia. Tällainen tilanne johtaa siihen, että asiakassovellus ohjaa käyttäjän kirjautumaan kertakirjautumisjärjestelmään, joka palauttaa

jo kirjautuneen käyttäjän asiakassovellukseen välittömästi oikeilla tiedoilla, mutta asiakassovellus palauttaa sen taas kertakirjautumisjärjestelmään, koska se ei osaa tallentaa kirjautumista oikein. Tästä tulee siis ikuinen uudelleenohjausketju, joka päättyy selainvirheeseen.

Rajapinnat

Rajapintaa määritettäessä pohdittiin eri toteutustapoja ja tultiin tulokseen, jossa Representational State Transfer (REST) -arkkitehtuurimalli olisi kaikkein lähestyttävin tapa toteuttaa järjestelmien välinen kommunikointi. Se oli ohjelmoijalle tuttu ja sisältäisi vähemmän työtä kuin esimerkiksi Extensible Markup Language (XML) -formaattiin perustuvat rajapinnat (Zazueta 2014).

Modulaarisen lähestymistavan vuoksi oli tärkeää hallita rajapinnan toiminnallisuutta ja yhteensopivuutta. Oli todennäköistä, että projekti laajenisi ajan myötä ennalta määräämättömään suuntaan sinäkin aikana, kun siihen liitetään useita asiakassovelluksia. Ei kuitenkaan haluttu rikkoa jo liitettyjen asiakassovellusten yhteensopivuutta, joten järjestelmän rajapinnat versioitiin. Näin asiakassovellukset voisivat sanoa suoraan rajapinnan osoitteessa käyttävänsä versiota X. Tämä mahdollistaa toiminnallisuuksien muuttamisen, kehittämisen ja jopa poistamisen, kunhan muistaa verhota muutokset tietyn version taakse. Käytettävä versio ilmoitettiin rajapintojen verkko-osoitteessa laittamalla sen alkuun versionumero muodossa v1, v2 tai v3, rsimerkiksi //sso/v1/login jne.

Kirjautumiseen vaaditut tunnisteet ja tiedot noudattavat taulukossa 1 kuvattua rajapintaa.

Taulukko 1. Rajapintakuvaus, jota asiakassovellukset noudattavat ohjatessaan käyttäjiä kirjautumaan kertakirjautumisjärjestelmällä.

Parametri osoitteessa	**Muoto**	**Selite**
app	Kokonaisluku	Sovelluksen tunniste
tid	Heksadesimaali	Epoch-aikaleima heksadesimaalina. Kysely vanhentuu 10 sekunnissa.
mode	"auth"	Kyselyn formaatti
sid	Alfanumeerinen	Yksilöi tunnistekyselyn. Palautetaan asiakassovellukselle sellaisenaan.
fields[]	Teksti	Kentät, joita käyttäjästä halutaan tietää. Vapaaehtoinen.
sig	Base64	Kyselyparametrien digitaalinen allekirjoitus RSA-avaimilla Base64-koodina.

		Kentät varmennetaan siinä järjestyksessä, kuin ne ovat tulleet.

Rajapinnan vaatimat tiedot lähetetään siis käyttäjän mukana uudelleenohjauksen GET-parametreina, eli osoitteen perässä esimerkiksi seuraavanlaisessa muodossa: ”http://kertakirjautumisjarjes-telma/sso/init?app=15&tid=53977&mode=auth&sid=A124&sig=YXNkZmzDtmtqYXNk-ZmzDt...”.

Onnistuneen kirjautumisen jälkeen kertakirjautumisjärjestelmä vastaa asiakassovellukselle taulukossa 2 kuvatun rajapinnan mukaan.

Taulukko 2. Onnistuneen kirjautumisen noudattelema rajapintakuvaus.

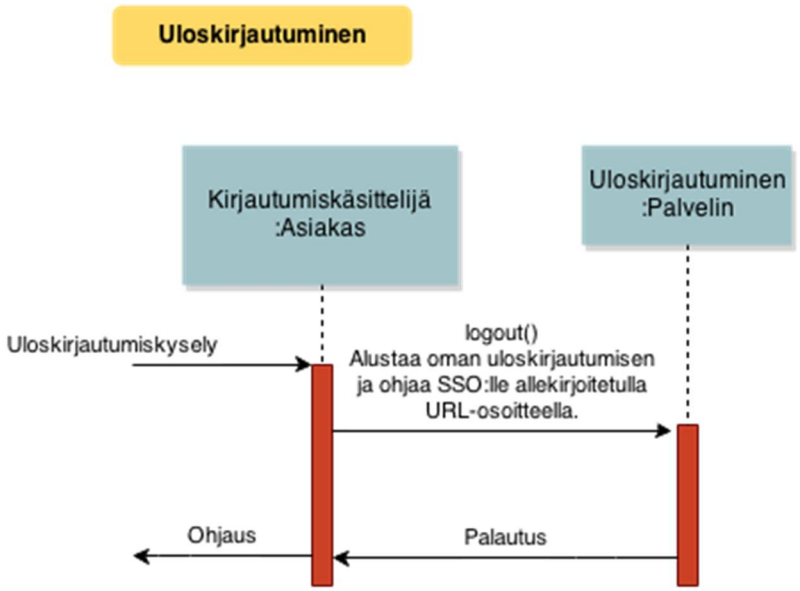
Parametri osoitteessa	**Muoto**	**Selite**
tid	Heksadesimaali	Epoch-aikaleima heksadesimaalina.
mode	"resp"	Kyselyn formaatti
sid	Alfanumeerinen	Asiakssovelluksen antama tunniste sellaisenaan
<kenttä>	Teksti	Kentät erillisinä parametreina, joita tunnistautumiskyselyssä pyydettiin. Mikäli ei pyydetä mitään, palautetaan "princ", "realname" ja "group[]"
sig	Base64	Kyselyparametrien digitaalinen allekirjoitus RSA-avaimilla Base64-koodina.

Nämä tiedot kulkevat asiakassovelluksen kirjautumiskäsittelijälle myös käyttäjän uudelleenohjauksen GET-parametreina. Oletuskentät ovat siis ”princ”, joka on käyttäjän yksilöivä tunniste, tässä tapauksessa sähköpostiosoite, ”realname”, joka kertoo käyttäjän oikean nimen ja tietyn määrän ”group[]”-kenttiä, jotka kertovat, missä asiakassovellukseen liitetyissä ryhmissä käyttäjä on, jos missään.

Uloskirjautuminen asiakassovelluksen kautta

Uloskirjautuminen asiakassovelluksen kautta noudattelee samaa kaavaa kuin tunnistautuminenkin. Asiakassovellus tarjoaa jonkin painikkeen, joka aloittaa uloskirjautumisprosessin, joka ohjaa käyttäjän tietyin parametrein kertakirjautumisjärjestelmän uloskirjautumisrajapintaan. Kyselystä tehdään samat tarkistukset kuin tunnistautumisessakin, mutta rajapinnan ”mode”-kenttä muutetaan ”auth”:sta ”logout”:ksi, mikä varmentaa, että

kyseessä on todella uloskirjautumisyritys. Kuvassa 11 näkyy, kuinka uloskirjautumisen kysely kulkee asiakassovelluksesta kertakirjautumisjärjestelmään ja takaisin.



Kuva 11. Uloskirjautuminen kertakirjautumisjärjestelmästä asiakassovelluksen välityksellä.

Onnistuneen uloskirjautumiskyselyn päätteeksi kertakirjautumisjärjestelmä palauttaa käyttäjän asiakassovelluksen palautusosoitteeseen samoilla tunnisteilla kuin tunnistautuessakin, mutta ilman käyttäjän tietoja. Tämän jälkeen asiakassovellus voi poistaa käyttäjän asiakassovelluskohtaiset istuntotiedot ja ohjata käyttäjän haluamaansa paikkaan. Mikäli istuntotietoja ei poisteta, asiakassovellukselle jäävät käyttäjän viimeisimmät tiedot voimaan eikä se toteutuksesta riippuen kierrätä käyttäjää seuraavalla kerralla kertakirjautumisjärjestelmän kautta.

5.3 Käyttöoikeudet

Järjestelmän sisäinen autorisointi suodattimilla

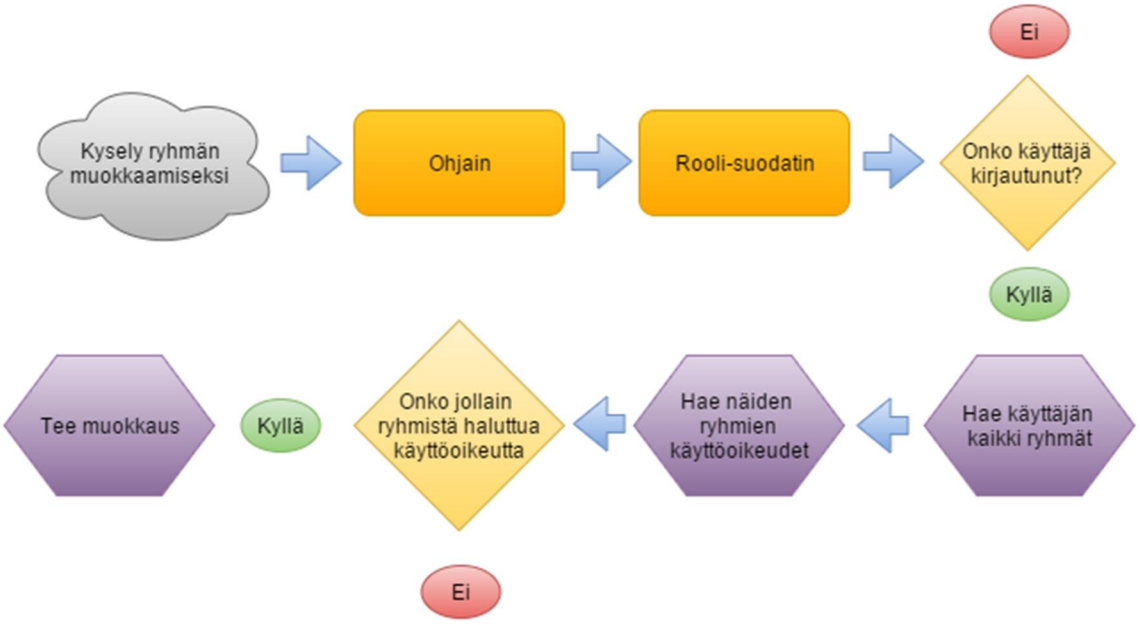
Kertakirjautumisjärjestelmä käyttää kaikkien kyselyjen autorisointiin reititysluokassa asetettuja suodattimia. Kun kysely tulee, se katsoo reititysluokasta kyselylle asetettuja käyttöoikeusvaatimuksia ja sen jälkeen katsoo, täyttääkö kyselyn tekijä asetetut vaatimukset.

Järjestelmän sisäinen autorisointimalli perustuu erikseen määriteltyihin käyttöoikeuksiin, joita annetaan tietyille ryhmille. Tämän jälkeen käyttäjiä liitetään asiaankuuluviin ryhmiin,

ja käyttöoikeudet olisivat näin käyttäjien hyödynnettävissä. Esimerkiksi ryhmien tietojen muokkaamiseen voisi olla oma käyttöoikeus ”Muokkaa ryhmiä”, joka olisi annettu ylläpitoryhmälle, johon kyselyn tekijän täytyisi kuulua. Käytännössä kuitenkin huomattiin, että käyttöoikeustarpeet olivat niiden ylläpitotyöhön nähden verrattain yksinkertaiset. Tämän vuoksi käytettiin vain kahta käyttöoikeutta, joita kutsuttiin rooleiksi, järjestelmäylläpitäjän rooli ja moderointirooli. Tietyille ryhmille annettiin järjestelmäylläpitäjän rooli, ja näissä ryhmissä olevat käyttäjät hallitsivat koko järjestelmää, ja moderointirooli annettiin ryhmille, jotka ylläpitivät vain niiden välittömässä ylläpidossa olevia asioita.

Käytännössä tehtiin autorisointimalli, jonka avulla voi olla monia käyttöoikeuksia, jotka on liitetty moniin ryhmiin, ja käyttäjiä, jotka saavat useita käyttöoikeuksia kuulumalla näihin ryhmiin.

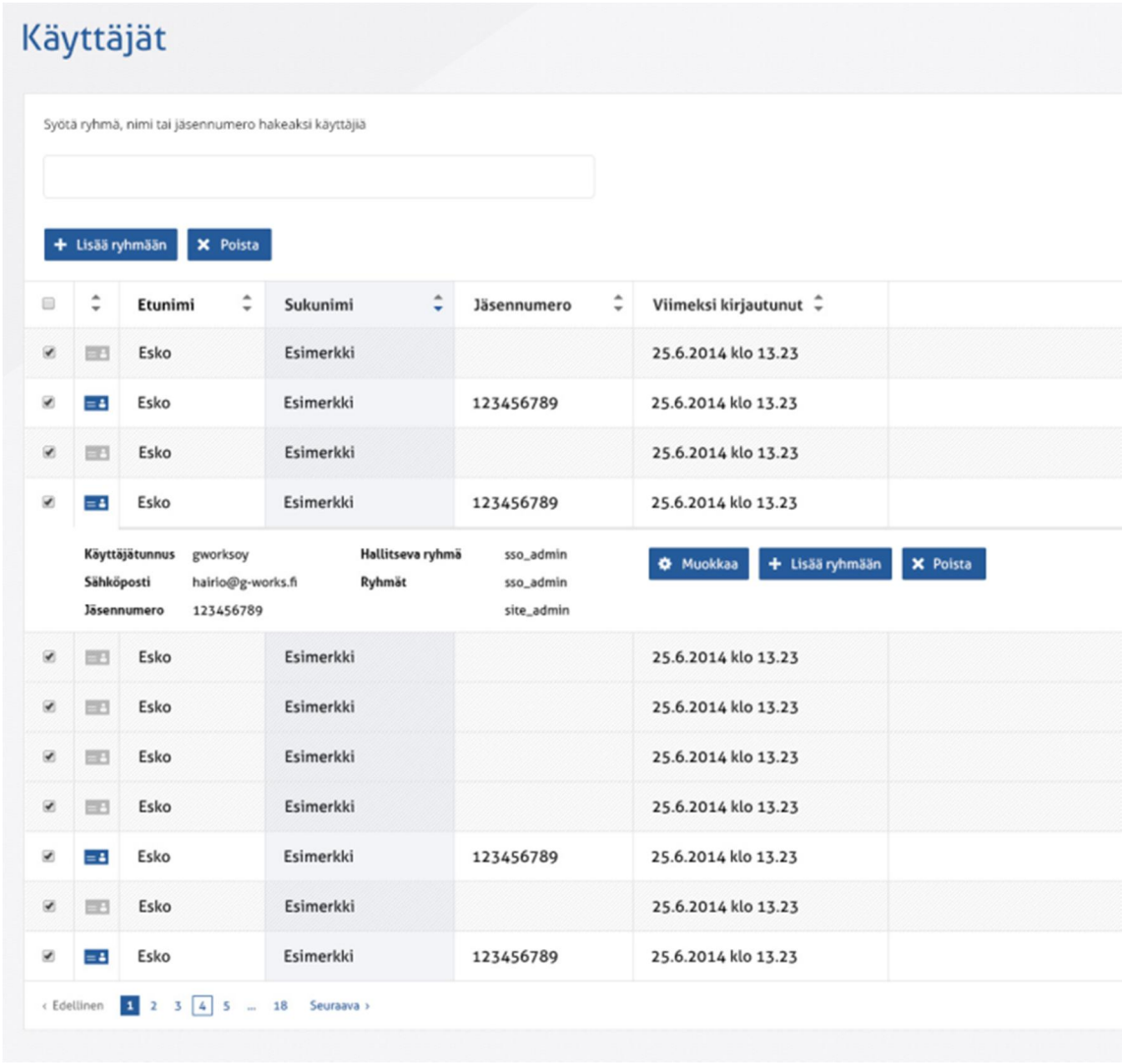
Kuvassa 12 on havainnollistettu kertakirjautumisjärjestelmän sisäinen tapahtumaketju, kun käyttäjän tekemiä toimia halutaan autorisoida kertakirjautumisjärjestelmän sisällä. Mikäli jokin näistä tarkistuksista ei tuota haluttua tulosta, käyttäjä palautetaan takaisin aikaisempaan näkymään virheviestin kera. Tarvittaessa tarkistuksia eli suodattimia voisi lisätä ohjaimessa rajattoman määrän kullekin toiminnolle, jolloin ne kaikki käytäisiin yksitellen läpi ja saataisiin varsin yksityiskohtainen autorisointimalli.



Kuva 12. Kertakirjautumisjärjestelmän sisäisten toimintojen autorisointitarkistus.

Käyttäjien hallinta

Kertakirjautumisjärjestelmälle luotiin oma hallintanäkymä, jossa hallittiin järjestelmän asetuksia, liitettyjä asiakassovelluksia, käyttöoikeusryhmiä ja niiden hierarkiaa sekä käyttäjiä, niiden tietoja ja käyttöoikeusryhmien liitoksia. Kuvassa 13 näkyy käyttäjien hakemiseen ja massahallintaan käytetty näkymä.



Kuva 13. Käyttäjähallinnan perusnäkymä, jossa voidaan etsiä ja hallita useita käyttäjiä yhtäaikaisesti.

Käyttäjien hallintanäkymässä keskeisenä käyttöliittymäelementtinä on käyttäjien hakukenttä. Haun käytettävyyttä ja skaalautuvuutta tuhansille käyttäjille pohdittiin, ja perinteisen syötteen ehdotuksen sijasta otettiin Fuse.js-hakukirjasto käyttöön. Fuse.js tekee approksimoitua tekstin kohdennusta (englanniksi Approximate string matching, Fuzzy

String Search). Tämän avulla hakusana voidaan syöttää vähän sinne päin tai hakea monesta kentästä samanaikaisesti, ja hakualgoritmi painottaa eri hakutuloksia sen mukaan, miten vahvasti ne osuvat annetun hakusanan merkkeihin (Risk). Esimerkiksi hakemalla ”Si Krnen” se voisi ehdottaa hakutulokseksi ”**S**am**i** **K**urvi**nen**” ja ”**S**ann**i** **K**er**a**n**en”**. Merkkien ei tarvitse olla oikeassa järjestyksessä, vaan se antaa niille painoarvoa sen mukaan, kuinka lähellä ne ovat oikeita merkkejä.

Fuse.js ei sovellu sellaisenaan suurien datamäärien hakuun suurista kenttämääristä, sillä se raportoi, että kahden kentän käsitteleminen 20 000 rivin datamassasta kestää noin yhden sekunnin (Risk). Jos datamäärä tästä vielä kasvaa, se muuttuu varsin hitaaksi käyttäjän näkökulmasta. Kertakirjautumisjärjestelmässä rivejä ei ole noin paljoa, mutta hakukenttiä on viisi. Hitaus ratkaistiin siten, että hakurivejä rajoitettiin kohdistamalla ensimmäiset kolme kirjainta hakusanasta kenttien kolmeen ensimmäiseen kirjaimeen ja tekemällä approksimoitua kohdennusta vain jäljelle jäävään datamäärään. Käytännössä tämä siis tarkoitti, että syötetyn hakusanan kolme ensimmäistä kirjainta täytyi olla jonkin hakukentän kolme ensimmäistä merkkiä. Hausta tuli riittävän nopea, eikä käyttäjäpalautteessa tullut moitetta.

Käyttäjien hallinnassa oli olennaista huolehtia käyttöoikeuksien tarkistuksista, jotta tiedon luottamuksellisuus säilyisi. Järjestelmäylläpitäjät saivat nähdä, lisätä ja hallita kaikkia kertakirjautumisjärjestelmän käyttäjiä, jopa muita järjestelmäylläpitäjiä. Alemman tason ylläpitäjät taas saivat nähdä vain omien ryhmien alaiset käyttäjät ja hallita niitä. Näiden lisäksi täytyi vielä tehdä tarkistuksia eri ryhmien näkyvyyksistä: ylläpitäjä ei saa nähdä muita järjestelmään luotuja ryhmiä kuin niitä, joita se itse hallitsee tai joihin se kuuluu, eikä se ei voi liittää muita käyttäjiä muihin kuin hallitsemiinsa ryhmiin.

Käytännössä käyttöoikeuksien tarkistaminen tehtiin aina tapauskohtaisesti, ennen kuin käyttäjiä listattiin hauissa, sallittiin käyttäjien muokkaaminen, liitettiin ryhmiin tai näytettiin käyttäjän ryhmiä jossain. Tämä siis tarkoittaa, että kun ylläpitäjä katselee käyttäjän ryhmiä, kaikista käyttäjän ryhmistä piilotetaan ne, joihin ylläpitäjällä ei ole oikeuksia, jolloin ainoastaan järjestelmäylläpitäjillä on kokonaiskuva käyttäjän ryhmistä ja käyttöoikeuksista.

Ryhmien hallinta

Käyttäjäryhmät ovat yksi kertakirjautumisjärjestelmän keskeisimmistä toiminnallisuuksista. Niiden perusteella asiakassovellukset antavat käyttäjille oikeuksia palveluihin, ja kertakirjautumisjärjestelmä määrittelee, hallitseeko käyttäjä muita käyttäjiä tai kertakirjautumisjärjestelmän asetuksia. Nämä hallintaoikeudet määrittyvät käyttäjien hallinnan osalta hierarkkisesti: jos on ylempänä kuin toinen, voi hallita alempana olevia, ja asetuksien osalta tietyille ryhmille annetaan erillisoikeus hallita niitä.

Kertakirjautumisjärjestelmän hallintapaneeliin luotiin oma käyttöliittymä ryhmien hallintaan. Ryhmät näytetään hierarkkisesti, mikä havainnollistaa käyttöoikeuksia ja kategorisointia, ja siinä hyödynnettiin samoja yleisiä käyttöliittymäperiaatteita kuin muissakin näkymissä: haussa käytettiin reaaliaikaista approksimoitua tekstin kohdennusta, taulukossa ryhmät pystyi järjestämään otsakkeen mukaan ja suuret tietomäärät sivutettiin.

Ryhmien hierarkkisuus toteutettiin yksinkertaisesti valitsemalla kullekin ryhmälle sen hallitseva ryhmä, joka syötti tämän viittauksen relaatiotietokantaan. Ryhmällä ei voi siis olla kuin yksi sitä hallitseva ryhmä. Asiakassovelluksien ja ryhmien liitokset toteutettiin omassa tietokantataulussa, joka ei ole hierarkkinen liitos, vaan tasavertainen monen suhde moneen -liitos. Tämä tarkoittaa, että yksi ryhmä voidaan liittää moneen asiakassovellukseen ja yhteen asiakassovellukseen voidaan liittää monta ryhmää.

Ryhmien hierarkkisuus ja niiden liitokset asiakassovelluksiin muodostavat mielenkiintoisen asetelman käyttöoikeuksien ja luottamuksellisuuden osalta. Kun käyttäjä kirjautuu asiakassovellukseen kertakirjautumisjärjestelmän kautta, asiakassovellukselle ei saa kertoa käyttäjän muista mahdollisista ryhmistä. Jos kerrottavia ryhmiä ei rajoiteta, saatetaan tahtomatta paljastaa asiakassovelluksille luottamuksellista tietoa käyttäjän asemasta, hierarkiasta tai toimenkuvasta organisaation sisällä. Samasta syystä toisia käyttäjiä hallitsevassa ryhmässä olevan käyttäjän näkemiä ryhmiä täytyy rajoittaa.

Kun on kyse tuhansista käyttäjistä ja kymmenistä tai sadoista hierarkkisista ryhmistä, ainoa tapa liittää käyttäjiä ryhmiin ei voi olla yksittäisen käyttäjän tasolla. Tästä syystä kertakirjautumisjärjestelmään luotiin massatyökaluja tekemään usein toistuvia toimia suurille käyttäjämäärille. Ryhmien osalta se tarkoitti käytännössä sitä, että ryhmän hallintanäkymästä pystyi kohdistamaan toimintoja joko kaikkiin ryhmän jäseniin tai vain erik-

seen valituille. Tällaisia toimia olivat esimerkiksi kaikkien ryhmän jäsenien liittäminen johonkin toiseen ryhmään, jolloin käyttöoikeuksien hallinnasta ei muodostu monivaiheista prosessia, tai kaikkien ryhmän jäsenien merkitseminen tunnistetuiksi käyttäjiksi.

5.4 Järjestelmän hallinta

Järjestelmän asetukset

Kertakirjautumisjärjestelmän asetukset ja määritykset asetettiin omalle sivulle hallintapaneelissa. Sivun näkyminen ja toimet sen alla asetettiin ”järjestelmäylläpitäjä”-suotimen alle, mikä tarkoittaa, että kaikki toimet siellä vaativat käyttäjän olevan järjestelmäylläpitäjäryhmässä. Asetuksissa hallitaan pääasiassa järjestelmän lähettämää sähköpostiliikennettä ja viestien sisältöjä sekä katsotaan järjestelmän lokimerkintöjä.

Sähköpostien hallinta oli verrattain yksinkertainen prosessi: kun järjestelmään tehtiin toiminnallisuus, joka lähetti sähköpostia, sille lisättiin hallintanäkymään uusi pohja sähköpostiviestin sisällöksi. Järjestelmäylläpitäjät pystyivät muokkaamaan tätä pohjaa, ottamaan sähköpostiviestin pois käytöstä tai valitsemaan, että järjestelmäylläpitäjille lähetettiin ilmoitus, kun jollekulle lähetettiin kyseinen sähköpostiviesti.

Sähköpostiviestipohjiin toteutettiin useista käyttöliittymien kuvauskielistä tuttu syntaksi (Mustache), jossa ”{{”- ja ”}}”-merkkien välissä oleva tunniste korvattiin asiaan kuuluvalla sisällöllä. Koodiesimerkissä 2 on esimerkki käyttäjälle lähetettävän aktivointisähköpostiviestin pohjasta, jossa ”{{”:n ja ”}}”:n sisällä olevat tunnisteet korvataan käyttäjän nimellä ja yksilöllisellä aktivointilinkillä.

```
Hei {{etunimi}} {{sukunimi}},

Olet luonut tunnukset kertakirjautumisjärjestelmään. Jotta
voit ottaa tunnukset käyttöösi, aktivoithan ne klikkaamalla
alla olevaa linkkiä.

{{aktivointi_linkki}}

Terveisin,
Organisaation yhteyshenkilö.
```

Koodiesimerkki 2. Aktivointisähköpostin pohja, jonka tunnisteosiot korvataan yksilöllisillä tiedoilla.

Koodiesimerkin 2 sähköpostipohjaa käytetään, kun käyttäjä on luonut itselleen tunnuksen kertakirjautumisjärjestelmään, ja sen aktivointia varten lähetetään sähköposti. Järjestelmä luo ensin käyttäjän annetuilla tiedoilla, luo sille yksilöllisen tunnisteen mutta asettaa tilin vielä aktivoimattomaan tilaan, jolloin sitä ei voi käyttää. Tämän jälkeen järjestelmä syöttää käyttäjän tiedot ja sähköpostipohjan tunnisteen tätä varten luotuun ”send_email_from_template”-metodiin. Tiedot syötetään ennalta määritetyssä formaatissa, joka kohdistaa pohjassa käytetyt tunnisteet oikeisiin tietoihin.

Send_email_from_template-metodi hakee halutun sähköpostipohjan asetuksista sen tunnisteen perusteella ja syöttää sen ja annetut tiedot ”render_template”-metodille. Render_template-metodi tekee varsinaisen käännöksen tunnisteista tiedoksi hyödyntämällä säännöllisiä lausekkeita. Käytännössä se siis poistaa ”{{”- ja ”}}”-merkit, ottaa niiden sisällä olevan tunnisteen ja korvaa sen sillä tunnisteella löytyvällä datalla. Kun sähköposti on luotu, se lähetetään käyttäjälle koodiesimerkissä 3 nähtävässä formaatissa.

```
Hei Esko Esimerkki,

Olet luonut tunnukset kertakirjautumisjärjestelmään. Jotta
voit ottaa tunnukset käyttöösi, aktivoithan ne klikkaamalla
alla olevaa linkkiä.

http://sso.organisaatio.fi/auth/activate/12356/98765

Terveisin,
Organisaation yhteyshenkilö.
```

Koodiesimerkki 3. Sähköpostipohjasta käännetty valmis sähköpostiviesti käyttäjätilin aktivointiin.

Tapahtumien lokitukset

Kertakirjautumisjärjestelmä kehitettiin vaiheittain, ja jokaisessa vaiheessa pyrittiin validoimaan järjestelmän toimintaa ja tarkoitusta. Haluttiin tietää, kuinka paljon järjestelmää käytetään, kuka sitä käyttää, miten sitä käytetään ja tuleeko sieltä jotain yllättäviä käyttötapauksia tai muuten hälyttäviä toimintoja, jotka toistuvat. Tästä syystä järjestelmään kehitettiin lokitoiminnallisuus, jota organisaation järjestelmäylläpitäjät pystyivät lukemaan. Lokitiedostot ovat myös varsin hyödyllisiä, kun pyritään paikantamaan mahdollisia virhetilanteita, väärinkäytöksiä tai tietoturvahyökkäyksiä (Sadowski 2010).

Lokijärjestelmästä nähtiin selkeästi eroteltuna, kuka on kirjautunut järjestelmään, kuka on kirjautunut mihinkin asiakassovellukseen, onko kierretty kertakirjautumisjärjestelmän

kautta jo kirjautuneena, kuka on muuttanut mitäkin tietojaan, kuka on tehnyt mitäkin ryhmäliitoksia tai minkätyyppisiä sähköpostiviestejä järjestelmä on lähettänyt kenellekin. Käytännössä siis kaikki järjestelmän hallintapaneelista tehdyt toimet jättävät jälkensä lokijärjestelmään.

Lokinäkymien ongelmaksi muodostuu helposti se, että siihen kerääntyy todella paljon tietoa, joka ei kuitenkaan ole kiinnostavaa kulloistakin tavoitetta varten. Tämä ongelma ratkaistiin sitomalla lokinäkymät aina johonkin kontekstiin. Esimerkiksi tiettyyn käyttäjään liittyvät lokit näkyvät käyttäjän profiilisivulla ja tiettyyn ryhmään liittyvät lokit ryhmän sivulla. Tällöin kaikki lokirivit ovat relevantteja siinä kontekstissa, kun niitä katsotaan. Tämän lisäksi jokaiseen lokinäkymään lisättiin suotimet, joiden avulla pystyy rajaamaan tietyntyyppisiä lokitapahtumia pois näkymästä.

Lokitapahtumat tallennettiin jokaisen tehdyn tapahtuman yhteydessä tietokantaan. Näille tapahtumille annettiin koneellinen nimi, esimerkiksi käyttäjän tietojen muokkauksen yhteydessä se olisi esimerkiksi ”user_edit.edit_information” tai luonnin yhteydessä ”user_edit.create_user”. Tämä yksinkertainen muoto mahdollistaa lokitietojen ohjelmallisen käsittelyn suoraan tietokannasta. Saatettiin esimerkiksi suodattaa tuloksia, yhdistellä ja laskea tulleita tapahtumia suoraan tietokannasta. Konekielinen ja lyhyt tapahtuman nimi ei ole kuitenkaan kovin käyttäjäystävällinen, eikä siitä välttämättä saa hyvin kuvaa siitä, mitä on oikeasti tapahtunut, jos ei osaa englantia tai ole muuten varsin hyvin perillä järjestelmän toiminnasta.

Konekieliset nimet voitiin kääntää Laravelin tarjoamalla käännöskirjastolla, kun tapahtumien nimiä käsiteltiin ihan tavallisen tekstisisällön tapaan. Kullekin tuetulle kielelle tehtiin /application/languages/-kansioon uusi kansio tuetun kielen lyhenteellä, esimerkiksi suomen kielessä ”fi”. Tähän kielikansioon tehtiin kielikäännöstiedosto, joka sisälsi PHP-koodina tavallisen taulukon, jossa avainsanoja vastaa selkokielinen kuvaus. Kuvassa 14 ovat sähköpostilokien konekieliset nimet ja niiden selkokieliset vastineet ”log.php”-nimisessä käännöstiedostossa.

```php
<?php
return array(
  'send_email' => array(
    'password_recover_notify' => 'Ilmoitus salasanapalautuksesta',
    'password_recover' => 'Salasanapalautus',
    'user_register_notify' => 'Ilmoitus käyttäjän rekisteröitymisestä',
    'user_register' => 'Käyttäjän rekisteröityminen',
    'user_activation' => 'Käyttäjän aktivointi',
    'user_activation_notify' => 'Ilmoitus käyttäjän aktivoinnista',
  )
);

```

Kuva 14. Laravelin kielikäännöstiedostot ovat PHP-taulukkoja.

Tätä käännöskirjastoa käytetään kutsumalla käännösfunktiota ja antamalla parametriksi käännöksen tunniste. Tunniste muodostetaan käännöstiedoston nimestä ilman päätettä ja käännöstaulukon soluista. Koodiesimerkissä 4 tulostetaan kuvassa 14 nähty selkokielinen selite salasanapalautuksen koneellisen nimen perusteella.

```
<?php
   echo __('log.send_email.password_recover');
?>


Tulostaa: "Salasanapalautus"
```

Koodiesimerkki 4. Käännösfunktion käyttö salasanapalautuksen sähköpostilokissa.

Lokijärjestelmän toinen tärkeä tehtävä oli erinäisten asioiden todentaminen. Kun selvitetään, mitä järjestelmässä on tapahtunut ja tehty, kun on tapahtunut jotain epäselvää, esimerkiksi käyttäjän tiedot ovat muuttuneet odottamatta, on kriittistä, että kaikki tapahtumaan liittyvät toimet voidaan todentaa tarkasti. Näissä tapauksissa ei riitä, että nähdään tietojen muuttuneen, vaan pitää pystyä todentamaan, kuka tietoja on muuttanut, mistä muutos on tehty, milloin se on tapahtunut, mistä lähtötilanteesta tiedot ovat muuttuneet ja millaiseksi ne ovat muuttuneet (Sadowski 2010).

Esimerkkitapaus todentamistarpeesta voisi olla seuraava: käyttäjä kirjautuu kertakirjautumisjärjestelmän kautta asiakassovellukseen, ja asiakassovellus näyttää sille väärää sukunimeä. Käyttäjä kirjautuu kertakirjautumisjärjestelmään ja näkee siellä sukunimensä olevan oikein ja kysyy järjestelmäylläpitäjiltä, miksi hänen sukunimensä on väärin asiakassovelluksessa. Järjestelmäylläpitäjä katsoo käyttäjän lokia ja näkee, että toinen ylläpitäjä on muuttanut vahingossa väärän henkilön sukunimeä ja korjannut sen hetken

päästä takaisin. Tässä välissä käyttäjä on kirjautunut asiakassovellukseen, joka on tallentanut väärän sukunimen, ja tilanne korjaantuu itsestään, kun asiakassovellus päivittää saamansa tiedot.

Jotta monivaiheisten toimien todentaminen olisi mahdollista riittävällä tasolla, tehdyt muutokset täytyy erottaa ja tallentaa erilleen muutoksen kohteista. Jos siis muutokset koskevat käyttäjää, lokijärjestelmä ei saa näyttää lokijärjestelmässä käyttäjän nykyisiä tietoja, vaan muutostenaikaisia tietoja, jotka on eriytetty esimerkiksi tietokannassa omaan paikkaansa. Tällä saavutetaan samalla se, että lokijärjestelmään voidaan tallentaa jo poistettujen tai täysin muuttuneiden käyttäjien tietoja tarkasti.

Riippuen lokijärjestelmään tallennettavan datan määrästä ja tyypistä se voi olla hyvinkin muuttuvaa, puhumattakaan, että järjestelmän kehittyessä tallennettavien asioiden määrä muuttuu ja kohteiden tietomallit vaihtelevat. Vaihtelevan rakenteen säilyttämiseksi käytetään usein NoSQL-tietokantoja perinteisten relaatiotietokantojen sijaan, sillä relaatiotietokantojen haasteena on niiden tarve määrittää tietorakenne hyvin tarkkaan (Vaish 2013: 7). Kertakirjautumisjärjestelmässä ei haluttu ottaa toista tietokantaa nykyisen relaatiotietokannan rinnalle, vaan ongelma ratkaistiin sarjallistamalla lokidata tietokantaan. Tieto sarjallistettiin PHP:n omalla ”serialize”-funktiolla.

Sarjallistetun tiedon syöttämisestä tietokantaan aiheutuu joitakin haittoja verrattuna tarkasti määriteltyyn ja syötettyyn tietoon. Sarjallistetusta tiedosta on merkittävästi hankalampi ja hitaampi tehdä hakuja, sarjallistettu tieto ei ole sellaisenaan helposti luettavissa tietokannasta, vaan se täytyy ottaa sieltä kokonaisuudessaan ja käsitellä ensin, ja tietokanta ei osaa optimoida tai indeksoida sarjallistettua tietoa (Tucker 2010). Nämä haitat eivät kuitenkaan häirinneet kertakirjautumisjärjestelmässä, sillä sarjallistettu lokitieto haettiin aina kokonaisuudessaan tietokannasta, sitä ei tarvinnut tehdä usein eikä sille ollut suorituskykypaineita.

Aputoiminnot

Ohjelmistokehityksessä tulee usein tilanteita, joissa huomataan, että joitain toimintoja tehdään toistuvasti ja ne tuntuvat työläiltä. Tämä on selvä merkki siitä, että toiminnot tulisi automatisoida mahdollisuuksien mukaan. Kertakirjautumisjärjestelmän kehityksessä näissä tilanteissa hyödynnettiin Laravelin tarjoamaa Artisan-nimistä komentorivityökalua.

Artisanilla pystyy käynnistämään Laravel-ohjelmistokehyksellä luodun sovelluksen, tässä tapauksessa kertakirjautumisjärjestelmän, komentoriviltä ja suorittamaan ohjelmakoodia siinä. Artisania käytetään joko itse luotujen tai valmiina olevien aputoimintojen suorittamiseen, kuten tietokantamigraatioiden, ajastettujen toimintojen tai yksikkötestien suorittamiseen (Laravel Tasks).

Kertakirjautumisjärjestelmän kaikki aputoiminnot liittyivät jollain tavalla tietokannan muokkaamiseen. Yleinen tietokantasovelluksien aputoiminto on ensimmäisten käyttäjätilien ja muun lähtötilanteen tiedon syöttö tietokantaan, ja tällainen tehtiin myös kertakirjautumisjärjestelmään. Toinen apuohjelma, joka luotiin, oli puhtaasti asiakasorganisaation toiveesta saada kertakirjautumisjärjestelmän käyttäjälista aktiivisista käyttäjistä. Lisäksi luotiin apuohjelmia olemassa olevan tiedon muokkaamiseen järjestelmän kehittyessä. Organisaatiolle tuli erään asiakassovelluksen myötä tarve listata kertakirjautumisjärjestelmässä henkilön organisaatiotunniste. Tämän toiminnallisuuden kehittämisen jälkeen huomattiin, että vanhojen käyttäjätilien organisaatiotunniste muuttui nollaksi, jos yritti muokata sellaista vanhaa käyttäjätiliä, jossa ei ollut asetettu mitään organisaatiotunnusta. Tämän korjaamiseksi luotiin pieni apuohjelma, jolla pystyttiin korjaamaan virheellisiä organisaatiotunnisteita sillä välin, kun saatiin vietyä tilanteen korjaava päivitys tuotantoon.

Istuntojen käsittely

HTTP-protokolla on määritykseltään täysin tilaton protokolla. Tämä tarkoittaa, että yksittäinen kysely ei tiedä muista kyselyistä mitään eikä ole protokollatasolla mahdollista todentaa, että jokin kysely olisi jonkin käyttäjän tekemä kysely tai että se olisi vielä kirjautuneena kyselyä tehdessä. Tämä ei kuitenkaan ole mitenkään poikkeuksellinen tarve, ja PHP-kielen tarjoamat istunnot pyrkivätkin ratkaisemaan tämän ongelman. Istunnoissa käyttäjälle luodaan yksilöllinen istuntoavain, jonka taakse tallennetaan järjestelmän tilaan liittyviä tietoa. Kun käyttäjä tarjoaa seuraavissa kyselyissä saman istuntoavaimen, palvelin osaa hakea samat tiedot, joita aikaisemmassa kyselyssä käytettiin, ja näin voidaan säilyttää käyttäjän ja sovelluksen tilaa tilattomassa HTTP-protokollassa.

Perinteinen tallennustapa tälle istuntoavaimelle on selaimen välimuistissa eli evästeissä. Selain tarjoaa kullekin verkko-osoitteelle oman välimuistin, jolloin www.organisaatio.fi:n välimuistiin asetettu kirjautumistieto ei sotkeennu www.toinenorganisaatio.fi:n välimuis-

tin kanssa (Barth, Jackson & Mitchell 2008: 78). Kertakirjautisjärjestelmän kehitysympäristö oli kuitenkin samassa verkko-osoitteessa organisaation toisen Laravelilla kehitetyn verkkosovelluksen kanssa, mikä aiheutti sen, että istuntoavaimet menivät näiden verkkopalveluiden osalta ristiin. Tämä johtui siitä, että molemmat verkkopalvelut käyttivät istuntoavaimen sijaintina Laravelin asettamaa välimuistin oletussijaintia. Tämä ratkaistiin muuttamalla Laravelin määrityksistä istuntoavaimen paikkaa välimuistissa kullekin palvelulle yksilölliseksi.

Kertakirjautumisjärjestelmän hallintaliittymän istunnot yritettiin päättää siihen, kun ylläpitäjä sulkee selaimensa, jotta vanhat hallintaistunnot eivät jää voimaan. Tämä oli aikaisemmin toteutettu jättämällä istunnon tuhoaminen käyttäjän selaimelle, mutta esimerkiksi Googlen Chrome-selain on versiosta 19 eteenpäin jättänyt istuntotiedot voimaan käytettävyyden nimissä (Peterson 2015). Tämän vuoksi ei voitu luottaa ylläpitäjien selaimiin, vaan jouduttiin itse kehittämään tarkistus istuntotietojen voimassaoloon, joka rajattiin yhteen tuntiin.

Laravel mahdollistaa istuntotietojen tallentamisen haluttuun paikkaan. Oletusarvoisesti se tallentaa istuntotiedot palvelimen levylle tiedostoon. Vaihtoehtoisina tapoina on tallentaa istuntotiedot tietokantaan, muistiin tai vaikka käyttäjän selaimen välimuistiin.

6 Järjestelmän käyttöönotto

6.1 Kertakirjautumisjärjestelmän ensimmäinen käyttöönotto

Kertakirjautumisjärjestelmä on itsessään hieman turha sovellus ilman asiakassovelluksia, joten se otettiin ensimmäisen kerran käyttöön yhdessä organisaation sisäisen Simple Machines Forum -keskustelupalstan kanssa. Kertakirjautumisjärjestelmälle luotiin liityntäluokka PHP-kielellä, jota hyödyntämällä asiakassovellukset pystyvät ottamaan kertakirjautumisjärjestelmän käyttöön mahdollisimman vähäisellä teknisellä toteutuksella. Käytännössä asiakassovelluksien vastuulle jäi liityntäluokan käyttöönotto sovelluksien eri vaiheissa.

Kertakirjautumisjärjestelmään tehtiin viimeiset testit ja korjaukset, jotka vietiin versionhallintaan. IT-osasto otti kertakirjautumisjärjestelmän versionhallinnasta, vei sen tuotantopalvelimelle ja asetti määrityksiin oikeat määritykset verkko-osoitteiden ja tietokantojen

osalta. Tämän jälkeen palvelimella suoritettiin aputoimintoina kehitetyt asennusskriptit, jotka loivat oikean tietokannan oikeilla käyttöoikeuksilla, loivat sinne oikean tietokantarakenteen, syöttivät ensimmäiset järjestelmäylläpitäjätunnukset ryhmineen ja sovelluksineen ja loivat oikeat RSA-avainparit.

Kun kertakirjautumisjärjestelmä oli itsessään toiminnassa, keskustelupalsta lisättiin kertakirjautumisjärjestelmään sovellukseksi, johon syötettiin sen julkinen avain tunnistamista varten. Käyttöönottoa helpottamaan tietyille avainhenkilöille ja osalle organisaation henkilöstöstä luotiin etukäteen tunnukset järjestelmään. Keskustelupalstaa varten luotiin omat ryhmät ja niiden kytkökset kertakirjautumisjärjestelmässä. Käyttäjät liitettiin asianomaisiin ryhmiin, jotta he voisivat kirjautua keskustelupalstalle.

Kun kertakirjautumisjärjestelmän toiminta ja liitos keskustelupalstaan oli varmennettu, kutsuttiin loppukäyttäjät mukaan. Organisaation henkilöstölle lähetettiin yksilöity sähköpostiviesti, joka sisälsi ohjeita keskustelupalstan käyttöön, ryhmähierarkiaan ja siihen, mitä oikeuksia niiden perusteella saa, sekä kertakirjautumisjärjestelmän käyttöohjeet ja käyttäjätunnukset.

Kertakirjautumisjärjestelmän toiminnallisuudet kehitettiin iteratiivisesti aina uusien asiakassovelluksien yhteydessä. Asiakassovellukset laajenivat keskustelupalstan liittämisestä erillisiin kommentointiliitännäisiin ja monisivusto-Wordpresseihin.

6.2 Liityntäluokka asiakassovelluksille

Aina kun kertakirjautumisjärjestelmään halutaan liittää uusi asiakassovellus, siihen liittyy määrättyjä toimenpiteitä. Kertakirjautumisjärjestelmä vaatii asiakassovelluksilta rajapinnan mukaisia tietoja, jotka tarjotaan aina samalla tavalla, joten oli luontevaa luoda tähän apukirjasto. Apukirjaston avulla asiakassovelluksien kehittäjät voivat keskittyä sovelluksen kehittämiseen ja käyttää apukirjastoa hoitamaan kertakirjautumisjärjestelmän liittäminen siihen.

Apukirjastoksi kehitettiin PHP-kielellä uusi liityntäluokka, jonka nimeksi tuli GwSSO (G-Works Single Sign-On). Liityntäluokan tarkoituksena on tarjota selkeä paikka asiakassovelluskohtaisille määrityksille, joiden perusteella se osaa hoitaa kommunikoinnin kerta-

kirjautumisjärjestelmän kanssa oikein muodostetun rajapinnan välityksellä. Liityntäluokkaa pystyi käyttämään sellaisenaan PHP-pohjaisissa asiakassovelluksissa, tai sitä pystyi käyttämään mallina toisille toteutuksille.

Kuvassa 15 on esimerkki, kuinka liityntäluokalla voidaan toteuttaa käyttäjän tunnistaminen kertakirjautumisjärjestelmän kautta. Kuvan koodikommenteista selviää kunkin osion tarkoitus.

```php
<?php
// Aloita istunto
session_start();

// Alusta liityntäluokka
require_once 'gwsso/gwsso.php';

// Sovelluskohtaiset määritykset
define('SSO_URL', 'http://sso.organisaatio.fi');
define('PASSPHRASE', 'yksityinen-salaustermi');
define('PATH_APPLICATION_PRIVKEY', 'polku-yksityiseen-avaimeen');
define('PATH_SSO_PUBKEY', 'polku-sson-julkiseen-avaimeen');
define('APPLICATION_ID', 1234);
define('REQUEST_FIELDS', []);

// Tietoa joka talletetaan ja säilötään tunnistautumisen välissä
$data = array(
    'redirect_url' => 'http://asiakassovellus.fi/profiili',
);

// Luo liityntäluokan instanssi
$sso = GwSSO::getInstance();

// Aseta määritykset luokalle
$sso->setSSOUrl(SSO_URL);
$sso->setupPrivateKey(PASSPHRASE, PATH_APPLICATION_PRIVKEY);
$sso->setupSSOKey(PATH_SSO_PUBKEY);
$sso->setData($data);

// Tunnista käyttäjä kertakirjautumisjärjestelmässä
$ssoReturn = $sso->auth(APPLICATION_ID, REQUEST_FIELDS);

if($ssoReturn !== 'AUTH_SUCCESSFUL') {
    die('Kertakirjautumisjärjestelmän käyttöönotossa tapahtui virhe.');
}

// Hae tunnistautuneen käyttäjän tiedot
$user = $sso->getUser();
$data = $sso->getData();

```

Kuva 15. Liityntäluokan käyttö käyttäjän tunnistautumisessa.

Kuvassa 15 koodirivi 31 on toteutuksen mielenkiintoisin rivi, sillä se tekee varsinaisen tunnistautumisen. Auth-metodi lopettaa ohjelman toiminnan ja ohjaa käyttäjän kertakirjautumisjärjestelmään. Kun käyttäjä on tunnistautunut siellä, hän palaa samaan koodiin,

jolloin Auth-metodi osaa lukea kertakirjautumisjärjestelmän palauttaman ”mode=resp” GET-parametrin. Tämän jälkeen se tarkastaa kaikki kertakirjautumisjärjestelmän lähettämät kentät. Kun tarkistus on tehty ja tunnistautuminen onnistunut, voidaan käyttäjän tiedot ottaa rivillä 38 näkyvällä komennolla. GetUser-metodi palauttaa käyttäjän tiedot rajapinnan mukaisesti, kuten sähköpostiosoitteen ja asiakassovelluksen ryhmät, joihin hän kuuluu. Rivillä 39 oleva getData-metodi palauttaa ennen tunnistusta rivillä 17 määritetyt tiedot, jotka voisivat olla dynaamisia ja käyttäjälle yksilöllisiä.

7 Yhteenveto

Insinöörityö ratkaisi asiakasorganisaation hajautetusta käyttäjähallinnasta koituneet hankaluudet luomalla kertakirjautumisjärjestelmän ja keskittämällä organisaation palvelut sen alle. Järjestelmä on ollut insinöörityön kirjoitushetkellä noin kolme vuotta käytössä, minkä aikana se on todettu toimivaksi. Se on asiakasorganisaation palautteen mukaan helpottanut käyttöoikeushallintaa merkittävästi, eikä vanhentuneita käyttöoikeuksia ole jäänyt järjestelmiin. Samoin asiakasorganisaatio on pystynyt reagoimaan yksittäisiin äkillisiin käyttöoikeusmuutoksiin pikaisesti.

Kertakirjautumisjärjestelmän toiminnan aikana ei ole raportoitu tai havaittu poikkeuksellisia tai hallitsemattomia vaikutuksia sen luottamuksellisuuteen, tiedon eheyteen tai saatavuuteen. Vuoden 2014 keväällä havaittu ”Heartbleed”-haavoittuvuus OpenSSL-järjestelmässä (Common Vulnerabilities and Exposures: CVE-2014-0160) olisi mahdollistanut luottamuksellisen tiedon varastamisen kertakirjautumisjärjestelmästä, mutta tähän reagoitiin riittävän nopeasti, eikä haavoittuvuutta ehditty hyödyntää. Kertakirjautumisjärjestelmä on siis todennettu riittävän tietoturvalliseksi.

Kertakirjautumisjärjestelmän käyttöönoton jälkeen siihen liitettyjen asiakassovelluksien käyttöönotto on ollut organisaation sisällä tavanomaista helpompaa ja suositumpaa organisaation oman palautteen perusteella. Kertakirjautumisjärjestelmä laajennettiin tukemaan myös organisaation ulkopuolisia käyttäjiä rajatuilla käyttöoikeuksilla, ja sitä on myös käytetty aktiivisesti eri asiakassovelluksissa.

Projekti täytti sille asetetut tavoitteet, ja järjestelmää pystyi helposti jatkokehittämään haluttuun suuntaan, mikä teki projektista onnistuneen. Haasteita toi kehitystiimin pienuus: kun tekninen kehitysvastuu on vain yhdellä henkilöllä tämän suuruusluokan projektissa,

siinä ei pääse ideoimaan eri toteutustapojen välillä ja haluttuun lopputulokseen pääseminen saattaa hidastua. Samoin projektin laajuus korosti ohjelmakoodin skaalautuvuuteen ja jäsentelyyn liittyviä haasteita. Järjestelmän sisäistä toteutustapaa täytyi kirjoittaa uudestaan ja toimintoja yleistää samaa tahtia, kuin järjestelmä kasvoi projektin aikana.

Kertakirjautumisjärjestelmä on yksi asiakasorganisaation keskeisimmistä järjestelmistä, ja se on liitoksissa lähes kaikkiin organisaation verkkopalveluista. Sitä voisi jatkokehittää tuotteistamalla sen missä tahansa organisaatiossa helposti käyttöönotettavaksi kertakirjautumisjärjestelmäksi.

Lähteet

Barth, Adam; Jackson, Collin; Mitchell, John C. 2008. CCS 08: Robust defences for cross-site request forgery. USA, New York: Association for Computing Machinery.

Common Vulnerabilities And Exposures: CVE-2014-0160. Verkkodokumentti. MITRE. <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160>. Luettu 6.3.2016.

Franco, Pedro. 2014. Understanding Bitcoin. Somerset: Wiley.

Gellman, Barton; Poitras, Laura. 2013. British intelligence mining data from nine U.S. Internet companies in broad secret program. Verkkodokumentti. <https://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html>. Luettu 11.3.2016.

Google Trends. Verkkodokumentti. Google. <https://www.google.fi/trends/explore#cat=0-5&q=%2Fm%2F0jwy148%2C%20%2Fm%2F09cjcl%2C%20%2Fm%2F09t3sp%2C%20%2Fm%2F02qgdkj&date=1%2F2012%2049m&cmpt=q&tz=Etc%2FGMT-2>. Luettu 17.3.2016.

Hope, Paco; Walther, Ben. 2008. Web Security Testing Cookbook. USA: Sebastopol.

Internet Engineering Task Force. 1999. RFC 2616. <https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>. Luettu 17.3.2016.

Kern, Christoph; Kesavan, Anita; Daswani, Neil. 2007. Foundations of Security: What Every Programmer Needs to Know. USA: Apress.

Laravel Tasks. Verkkodokumentti. Laravel. <http://laravel3.veliovgroup.com/docs/artisan/tasks>. Luettu 11.3.2016.

Mustache. <https://mustache.github.io/>. Luettu 11.3.2016.

OWASP Top 10 – 2013. 2013. Verkkodokumentti. The Open Web Application Security Project. <https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project> . Luettu 1.3.2016.

Peterson, Dave. 2015. Session Cookies in Chrome, Firefox, and Sitecore. Verkkodokumentti. <http://blog.petersondave.com/cookies/Session-Cookies-in-Chrome-Firefox-and-Sitecore/>. Luettu 17.3.2016.

Rees, Dayle. 2012. Laravel: Code Happy. Leanpub.

Risk, Kiro. Fuse.js. Verkkodokumentti. <http://kiro.me/projects/fuse.html>. Luettu 7.3.2016.

Rousku, Kimmo. 2014. Kyberturvaopas. Helsinki: Talentum.

Sadowski, Gorka. 2010. Using logs for forensics after a data breach. Verkkodokumentti. <http://www.networkworld.com/article/2193990/tech-primers/using-logs-for-forensics-after-a-data-breach.html>. Luettu 17.3.2016.

Semantic Versioning. Verkkodokumentti. OSGi Alliance. <https://www.osgi.org/developer/white-papers>. Luettu 6.3.2016.

Shiflett, Chris. PHP Security Guide. Verkkodokumentti. <http://phpsec.org/projects/guide/1.html#1.1>. Luettu 17.3.2016.

SSO Benefits. Verkkodokumentti. University of Guelphin. <https://www.uoguelph.ca/ccs/security/internet/single-sign-sso/benefits>. Luettu 1.3.2016.

Stevens, M.M.J. 2007. On Collisions for MD5. Eindhoven: Eindhoven University of Technology.

Stuttard, Dafydd; Pinto, Marcus. 2011. The Web Application Hacker's Handbook. USA Indianapolis, Indiana: John Wiley & Sons.

Top 9 Browsers on Feb 2016. Verkkodokumentti. StatCounter Global Stats. <http://gs.statcounter.com/#all-browser-ww-monthly-201602-201602-bar>. Luettu 17.3.2016.

Tucker, Morgan. 2010. When should you store serialized objects in the database? Verkkodokumentti. <https://www.percona.com/blog/2010/01/21/when-should-you-store-serialized-objects-in-the-database/>. Luettu 17.3.2016.

Vaish, Gaurav. 2013. Getting Started With NoSQL. UK Birmingham: Packt Publishing.

Wu, Hanqing; Zhao, Liz. 2015. Web Security: A WhiteHat Perspective. Boca Raton, FL, USA: CRC Press.

Zalewski, Michael. 2011. Tangled Web: A Guide to Securing Modern Web Applications. San Francisco, CA, USA: No Starch Press.

Zazueta, Rob. 2014. API Data Exchange: XML vs. JSON. Verkkodokumentti. <http://www.mashery.com/blog/api-data-exchange-xml-vs-json>. Luettu 17.3.2016.