TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

PÄIVI HIETANEN

SÄHKÖISEN ASIOINNIN WEBLOKIEN HYÖDYNTÄMINEN PÄÄTÖKSENTEOSSA

Diplomityö

Tarkastaja: tutkijatohtori Outi Sievi-Korte
Tarkastaja ja aihe hyväksytty
31. 05. 2017

TIIVISTELMÄ

Päivi Hietanen: Sähköisen asioinnin weblokien hyödyntäminen päätöksenteossa
Tampereen teknillinen yliopisto
Diplomityö, 51 sivua
Toukokuu 2018
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Data Engineering
Tarkastaja: tutkijatohtori Outi Sievi-Korte

Avainsanat: päätöksenteko, webloki, tiedonlouhinta verkosta, käytön louhinta verkosta, Elastic-ohjelmistopino

Päätöksenteko kuuluu oleellisena osana organisaatioiden jokapäiväiseen toimintaan. Jotta päätöksenteossa onnistutaan, tarvitsevat päätöksentekijät asianmukaista informaatiota päätöksenteon eri vaiheissa. Valtaosa tarvittavasta informaatiosta on olemassa organisaatioiden omissa perusjärjestelmissä, tarvitaan vain välineitä tiedon hyödyntämiseen. Weblokitietoja voidaan hyödyntää ihmisen ja tietojärjestelmän välisen vuorovaikutuksen tutkimisessa sekä tietojärjestelmien toiminnan analysoinnissa.

Tämän työn tavoitteena oli selvittää, miten weblokeja voidaan hyödyntää päätöksenteossa. Tavoitetta lähestyttiin tapaustutkimuksen avulla toteuttamalla tilaajaorganisaatiolle järjestelmä, jonka avulla sähköisen asiointipalvelun lokitiedostoista voidaan tuottaa informaatiota päätöksenteon tueksi. Toteutetun järjestelmän avulla kerättyä informaatiota tarkasteltiin eri organisaatioyksiköiden päätöksentekijöiden kanssa.

Työn käytännön osuuden toteutuksessa hyödynnettiin Elastic-ohjelmistopinoa ja tiedonlouhinnan menetelmiä. Tiedonlouhinnan menetelmäksi valikoitui käytön louhinta verkosta, joka käyttää tietoaineistonaan weblokeja. Työn tuloksena tilaajaorganisaatiolle toteutettiin järjestelmä, jonka kautta sähköisen asiointipalvelu analysointi on mahdollista. Järjestelmän avulla palvelun käytöstä ja toiminnasta louhittiin informaatiota, jota voidaan hyödyntää päätöksenteossa. Informaatio tarjoaa tukea päätöksiin, jotka liittyvät asiointipalvelun kehittämiseen tai siihen liittyviin resursointeihin.

ABSTRACT

Päivi Hietanen: Utilization of eService weblogs in decision making
Tampere University of Technology
Master of Science Thesis, 51 pages
May 2018
Master's Degree Programme in Information Technology
Major: Data Engineering
Examiner: Postdoctoral researcher Outi Sievi-Korte

Keywords: decision making, web log, web mining, web usage mining, Elastic-stack

Decision making is an important part of the everyday activities in an organization. To succeed in decision making, relevant information is needed to assist decision makers in the different phases of decision making. Majority of this information can be found in the organizations basic information systems. What is needed is the tools to utilize this information. Weblog information can be used to study human-computer interactions and to analyze performance of a computer system.

The objective of this master's thesis was to find out how weblogs can be utilized in decision making. To accomplish this objective a case study was conducted. In the case study, a system was implemented for the subscriber organization. The system produces information from the organizations eService logs. The information gathered though the implemented system was examined together with the decisionmakers from different organization units.

Elastic-stack and data mining methods were used in the implementation of this project. Web usage mining was selected to implement data mining, since it utilizes weblogs. As a result of this project a system was built that can be used to analyze user behavior and application performance. The system provides information that can be used to support decision makers in decisions concerning eService development or resource planning.

ALKUSANAT

Tämä diplomityö on toteutettu työnantajalleni, jota haluan kiittää mielenkiitoisen ja haastavavan aiheen tarjoamisesta. Kiitokset myös työkavereilleni, jotka tukivat työn toteutuksessa ja auttoivat eteen tulleiden haasteiden selvittelyssä.

Työn ohjauksesta haluan kiittää diplomityön tarkastajaa FT Outi Sievi-Kortetta. Erityisesti häneltä saamani palaute ja kommentit olivat arvokkaita työn eri vaiheissa.

Haluan kiittää perhettäni opiskelujeni tukemisesta. Erityiskiitokset puolisolleni, joka jaksoi kannustaa ja tukea pitkän diplomityöprojektin aikana. Hänen apunsa työn viimeistelyssä oli korvaamatonta.

Hämeenlinnassa, 17.05.2018

Päivi Hietanen

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

BI	Business Intelligence
CRUD	Create, Read, Update, Delete
DSL	Domain Specific Language
DSS	Decision Support System
ETL	Extract-Transform-Load
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
OLAP	OnLine Analytical Processing
REST	Representational State Transfer
SPA	Single-Page Application
SQL	Structured Query Language
URL	Unifor Resource Locator
W3C	World Wide Web Consortium

1. JOHDANTO

Organisaatiot tarvitsevat informaatiota päätöksenteon tueksi. Tarvittavasta informaatiosta suurin osa on jo olemassa organisaation omissa perusjärjestelmissä. Onnistunut päätöksenteko organisaatiossa edellyttää tämän informaation hyödyntämistä. Päätöksenteon tukemiseen on olemassa räätälöityjä tietojärjestelmiä, joiden avulla asianmukainen informaatio saadaan kootusti päätöksentekijöiden käyttöön.

Organisaatioiden operatiiviset tietojärjestelmät kirjaavat tapahtumiaan lokitiedostoihin. Näitä lokimerkintöjä voidaan hyödyntää käyttäjien ja tietojärjestelmien vuorovaikutuksen tutkimisessa sekä tietojärjestelmien toiminnan analysoinnissa. Lokeihin tallentuu päivittäin valtavia määriä erilaisia tapahtumia. Tiedonlouhinnan avulla suurista tietoaineistoista voidaan löytää organisaatiota hyödyttävää informaatiota.

Tämän työn tutkimuskysymys: ”Miten weblokeja voidaan hyödyntää päätöksenteossa?” syntyi tilaajaorganisaation tarpeesta tuottaa webpalvelustaan informaatiota päätöksenteon tueksi. Tutkimuskysymystä lähdettiin selvittämään tapaustutkimuksen avulla toteuttamalla tilaajaorganisaatiolle järjestelmä, jonka avulla sähköisen asiointipalvelun lokitiedostoista pystytään tuottamaan informaatiota päätöksenteon tueksi. Tiedonlouhinnan avulla pyrittiin tuottamaan informaatiota palvelun käytöstä ja käyttäjistä sekä asiointipalvelusovelluksen toiminnasta. Järjestelmän avulla tavoitteena oli saada tietoa palvelun käytöstä yleisesti: mihin palvelua käytetään ja ketkä palvelua käyttävät. Palvelussa olevista lomakkeista haluttiin tarkempaa tietoa lomakkeiden täyttöaikoihin ja mahdollisiin ongelmakohtiin liittyen. Tiedonlouhinnan tulokset oli tarkoitus esittää päätöksentekijöille visualisoituina, mahdollisesti helppokäyttöisen käyttöliittymän kautta, jolloin jokainen päätöksentekijä pystyisi itse etsimään tietoa omiin tarpeisiinsa.

Tutkielma koostuu kuudesta luvusta. Luku 2 esittelee organisaatioiden päätöksentekoon liittyvää teoriaa. Luvussa 3 käydään läpi lokien käyttötarkoitusta ja sitä, mitä pitää huomioida, kun lokitietoja hyödynnetään tiedonlouhinnassa. Luku 4 käsittelee tiedonlouhintaa yleisesti ja kappaleessa 4.4 esitellään työn toteutuksessa käytetty tiedonlouhintaprosessi. Luvussa 5 kuvataan yleisellä tasolla tilaajaorganisaation sähköinen asiointipalvelu ja käydään läpi työn tekninen toteutus. Luku 6 kuvaa tarkemmin työssä toteutetun tiedonlouhintaprosessin ja tiedonlouhinnan tulokset. Lopuksi kappaleessa 7 esitellään työn johtopäätökset ja jatkokehitysmahdollisuudet. Tutkielman keskeiset käsitteet on lihavoitu.

2. PÄÄTÖKSENTEKO ORGANISAATIOSSA

Päätöksenteko kuuluu oleellisena osana organisaatioiden toimintaan. Organisaatioissa tehdään päätöksiä päivittäin. Päätökset voivat vaihdella vähäpätöisistä merkittäviin ja ne voivat koskettaa yhtä tai useampaa henkilöä. Tässä kappaleessa käydään läpi päätöksenteon teoriaa organisaatioiden näkökulmasta ja sitä, miten tietojärjestelmät pystyvät tarjoamaan tukea organisaatioiden päätöksentekoprosessiin.

2.1 Päätöksenteko

Päätöksentekoa tutkitaan useissa tieteenaloissa: käyttäytymistieteissä, filosofiassa, psykologiassa, sosiologiassa, matematiikassa, taloustieteissä, hallintotieteissä ja tekniikan alalla [23; 30]. **Päätöksenteolla** tarkoitetaan päätöstä eri vaihtoehtojen välillä. Päätös on valinta, jonka vaikutuksesta jokin vaihtoehdoista hyväksytään toteutettavaksi. Päätöksenteko on myös resurssien kohdentamista. Paloheimo ja Wiberg määrittävät kirjassaan [29] **päätöksentekijäksi** henkilön, jolla on käskyvalta kohdennettavista resursseista. Päätöksentekijä valitsee vaihtoehdoista sopivimman puntaroimalla vaihtoehtoja valintakriteerien avulla. [23; 29; 30; 32]

Päätöstentekijöiden lukumäärän mukaan päätöksenteko voidaan jakaa yksilölliseen, interaktiiviseen ja kollektiiviseen päätöksentekoon [29]. Näistä kollektiivista päätöksentekoa tehdään erilaisissa organisaatioissa, joissa päätöksentekoa koskee yhteisesti sovitut yhteiset normit, tavat ja säännöt. Päätöksentekoa on tutkittu erilaisten mallien avulla. Seuraavassa kappaleessa kerrotaan päätöksenteon malleista liiketoiminnan näkökulmasta.

2.2 Päätöksenteon mallit

Kirjallisuudessa yleisemmin esiintyviä päätöksenteon malleja ovat rationaalinen, rajoitetusti rationaalinen ja inkrementaalinen. Rationaalinen päätöksenteko painottaa päätöksenteon tavoitteellisuutta, tehokkuutta ja kokonaisvaltaisuutta [31]. Siinä etsitään sopivat keinot tavoitteisiin pääsemiseksi. Liiketoiminnassa päätöksenteon tarkoitus on edistää yrityksen tavoitteiden saavuttamista, jolloin päätösprosesseissa pyritään valitsemaan ne vaihtoehdot, joiden katsotaan olevan asianmukaisia asetettujen tavoitteiden saavuttamiseksi [23; 34]. Päätöksenteko on kuitenkin harvoin rationaalista, sillä päätöksenteon rationaalisuutta rajoittavat psykologiset ja tiedolliset tekijät. Päätöksentekijöillä on usein rajalliset kyvyt identifioida ja arvioida eri vaihtoehtoja, heillä ei myöskään aina ole saatavilla kaikkea tietoa kaikista mahdollisista vaihtoehdoista [29; 31]. Päätöksentekijät ovat myös usein haluttomia tekemään rohkeita muutoksia ympäristöön päätöksiensä kautta.

He suosivat päätöksiä, jotka luovat inkrementaaleja muutoksia tavoitteiden uudelleen muotoilun ja kompromissien kautta. [31; 32]

Liiketoiminnassa tehdään päätöksiä useiden eri päätöksentekomallien mukaisesti. Pyrkimys päätöksenteossa on noudattaa rationaalista päätöksenteon mallia tehden päätöksiä, jotka tukevat asetettujen liiketoimintatavoitteiden saavuttamista. Päätöksentekijään kohdistuu erilaisia rajoitteita, jolloin päätöksenteossa noudatetaan useimmiten rajoitettua rationaalista tai inkrementaalista päätöksenteon mallia.

Päätöksentekomallien eroja voidaan havainnollistaa esimerkin avulla. Kuvitellaan tilanne, jossa uuden ohjelmiston kehitysprosessin suunnitteluvaiheessa tehdään päätös ohjelmiston arkkitehtuuriratkaisuista. Inkrementaalinen päätöksenteko pyrkii valitsemaan arkkitehtuuriratkaisuista sen, joka on tutuin ja jota on jo mahdollisesti organisaatiossa käytetty, vaikka se ei olisikaan soveltuvin ko. projektiin. Rajoitettu rationaalinen päätöksenteko pyrkii tarkastelemaan arkkitehtuuriratkaisuja rationaalisesti, mutta päätöksentekijöiden henkilökohtaiset mieltymykset eri arkkitehtuuriratkaisuiden välillä voivat painaa enemmän kuin ratkaisun toimivuus ko. ohjelmistoprojektissa. Rationaalinen päätöksenteko tarkastelee eri arkkitehtuuriratkaisuja ja valitsee niistä projektin tavoitteiden kannalta parhaimman.

2.3 Päätöksenteon vaiheet

Päätöksentekoon liittyy eri vaiheita. Paloheimo ja Wiberg [29] jakavat kollektiivisen päätöksenteon organisaatioiden yhteydessä seuraaviin vaiheisiin:

1. *Aloite*. Ongelma tai mahdollisuus havaitaan, tehdään aloite asiaa koskevan päätöksenteon käynnistämiseksi.
2. *Ongelman muotoilu*. Eri toimijat yrittävät jäsentää ongelmaa tai mahdollisuutta omista lähtökohdistaan.
3. *Vaihtoehtojen selvittäminen.* Täsmennetään päätöksenteon tavoitteet, tarpeet ja selvitetään käytettävissä olevat resurssit. Tavoitteena on esittää erilaisia päätösvaihtoehtoja, sekä arvioida vaihtoehtojen hyödyt, haitat, kustannukset, rajoitteet ja toteuttamistodennäköisyydet.
4. *Vaihtoehtojen arviointi ja päätöksen tekeminen.* Päätöksentekijä vertailee eri vaihtoehtoja ja valitsee niistä jonkin.
5. *Päätöksen täytäntöönpano.*
6. *Evaluaatio ja palaute.* Päätöksentekoa arvioidaan täytäntöönpanon aikana ja sen jälkeen. Evaluaation tärkein tehtävä on verrata päätöksenteon tuloksia ja vaikutuksia asetettuihin tavoitteisiin. Evaluaatioon voi liittyä palautteen hankkiminen heiltä, joita päätöksenteko koskee.

Simon jakaa [33] päätöksenteon neljään eri vaiheeseen. Vaikka vaiheita on vähemmän, ne sisältävät pääpiirteissään samat toimenpiteet kuin edellä kuvattu Paloheimon ja Wigbergin jaottelu.

Esimerkkinä päätöksentekoprosessista voidaan kuvata hankintapäätöksen kulku:

1. Havaitaan ongelma: organisaatiolle kriittinen laite on mennyt rikki.
2. Organisaation eri yksiköt jäsentelevät ongelmaa omasta näkökulmastaan. Kuinka kauan yksikkö pärjätäänkö ilman laitetta? Onko laitteen korjaaminen yksikön kannalta järkevää? Mitkä ominaisuudet ovat yksikön kannalta oleellisia, jos uusi laite päätetään hankkia?
3. Selvitetään eri vaihtoehdot. Korjataanko laite vai hankitaanko uusi? Mitkä ovat eri vaihtoehtojen hyödyt ja haitat? Korjaustarveselvityksen tekeminen ja tarjouspyyntö korjauksesta. Minkälainen laite hankitaan? Hankinnan tarjouspyyntöjen tekeminen. Määritetään päätöksenteon tavoitteet ja tarpeet, sekä käytettävissä olevat resurssit.
4. Arvioidaan eri vaihtoehdot (vastineet tarjouspyyntöihin) ja tehdään hankintapäätös tai korjauspäätös.
5. Laitehankinnan tai korjauksen toimeksianto ja sopimuksen vahvistaminen ehtoineen.
6. Seurataan hankitun laitteen toimivuutta ja kerätään palautetta laitteen käyttäjiltä. Laitteen toimivuutta ja saatua palautetta evaluoidaan hankinnalle asetettuihin tavoitteisiin ja sopimusehtoihin.

2.4 Päätöksenteon tukeminen

Organisaatioiden eri tahot tarvitsevat informaatiota päätöksenteon tueksi ja valtaosa tarvittavasta tiedosta on olemassa organisaation omissa perusjärjestelmissä [16]. Onnistunut päätöksenteko edellyttää oleellisen ja asianmukaisen informaation hyödyntämistä. Päätöksentekoa voidaan tukea olemassa olevalla historiatiedolla tai tietoa voidaan erityisesti kerätä tukemaan yksittäistä päätöksentekoprosessia. [32]

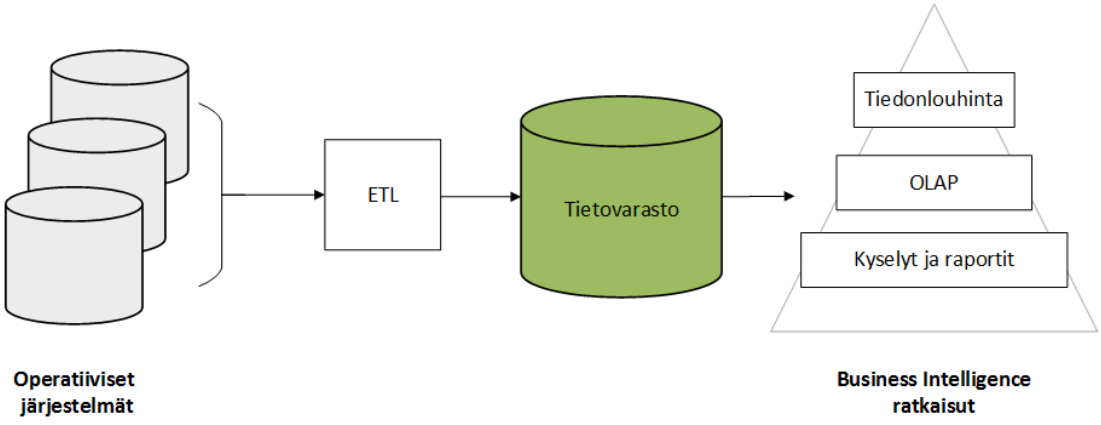
Sauter kertoo kirjassaan [32] tietojärjestelmien antavan tukea useaan päätöksenteon vaiheeseen. Tietojärjestelmät tarjoavat asianmukaista informaatiota ja malleja, joiden avulla päätöksentekijä voi tarkastella ongelmaan tai mahdollisuutta useammasta näkökulmasta. Päätöksenteossa eri vaihtoehtoihin liittyvän informaation organisointia ja tarkastelua kutsutaan mallintamiseksi. Ensimmäisessä päätöksenteon vaiheessa, jossa päätöksentekijä tunnistaa ongelman tai mahdollisuuden, tietojärjestelmät auttavat toimintaympäristön tarkkailussa ja organisaation tehokkuuden arvioimisessa. Seuraavissa vaiheissa, joissa ongelma muotoillaan ja vaihtoehdot selvitetään, tietojärjestelmät tarjoavat apua tiedon

kokoamiseen ja mallintamiseen. Lopuksi, kun eri vaihtoehdot arvioidaan ja päätös tehdään, voi päätöksentekijä tietojärjestelmien avulla käsitellä koottua informaatiota, vertailla eri vaihtoehtoja ja valita vaihtoehdoista parhaan. [32]

Päätöksenteon ja tietojärjestelmien yhteydessä käytetään usein termiä Business Intelligence (BI). Business Intelligence -termi kattaa kokoelman käsitteitä ja menetelmiä, joilla liiketoiminnan päätöksentekoa voidaan jalostaa [20]. Useimmiten BI-termiä käytetään kuvaamaan tiedon hyväksikäyttöä liiketoiminnassa. BI-ratkaisujen tavoitteena on tuottaa informaatiota päätöksenteon tueksi. Informaation avulla päätöksentekijät ymmärtävät organisaation toimintaa paremmin, mikä auttaa organisaatiota tekemään parempia päätöksiä ja ohjaamaan toimintaa haluttuun suuntaan. [16]

BI-ratkaisut perustuvat tiedon integrointiin ja tietovarastointimenetelmiin. Tietovarastot toimivat pohjana oleellisen liiketoimintatiedon hallinnalle. Varsinaiset BI-ratkaisut keskittyvät informaation analysoimiseen, esittämiseen ja jakeluun. [16] Kopácková esittelee artikkelissaan [20] yleisesti hyväksytyn mallin BI-komponenteista. Siinä tietovaraston päällä on kolme kerrosta: kyselyt ja raportit, OLAP (Online Analytical Processing) ja tiedonlouhinta (Kuva 1).

Business Intelligence -ratkaisujen tavoitteena on nopeuttaa ja parantaa organisaatioiden päätöksentekokykyä, vastata käyttäjien tietotarpeisiin oikea-aikaisesti, tukea organisaation strategiaa ja tavoitteisiin pääsyä, parantaa käyttäjien omatoimisuutta tietotarpeiden suhteen sekä vähentää kustannuksia ja parantaa operatiivista tehokkuutta. Parhaimmillaan BI-ratkaisut voivat nostaa päätöksentekijöiden käyttöön tietojärjestelmien syvyyksissä piilevää informaatiota. [16]



Kuva 1. *Business Intelligencen tietovirta ja komponentit*

Tietojärjestelmien avulla voidaan päätöksenteon tueksi tarjota reaaliaikaista, useasta eri lähteestä yhdistettyä ja analysoitua informaatiota. Tietojärjestelmien kehittyessä ja tallennustilan kustannusten laskiessa yhä suurempia tietomääriä voidaan hyödyntää päätöksenteossa.

3. WEBLOKIT JA LOKITIEDON HYÖDYNTÄMINEN

Loki on aikajärjestyksessä kirjattu tallenne tapahtumista. Tietojärjestelmät ovat kirjanneet tapahtumiaan lokitiedostoihin 1980-luvulta lähtien. Lokityyppejä on monenlaisia ja niillä on omat käyttötarkoituksensa, käyttötapansa ja muotonsa. [28; 39]

Tässä kappaleessa käydään läpi mitä weblokilla tarkoitetaan, mitkä ovat weblokien käyttötarkoitukset ja mitä pitää huomioida, kun weblokeilla olevaan tietoa hyödynnetään päätöksenteossa.

3.1 Webloki ja sen käyttötarkoitukset

Webloki sijaitsee tyypillisesti WWW-palvelimella, välityspalvelimella tai käyttäjän selaimessa. Weblokeille kirjataan tietoa, aina kun käyttäjä tekee pyynnön WWW-palvelimelle. [27; 38] WWW-palvelimella voi WWW-palvelinohjelmiston lokien lisäksi olla myös yksittäisten verkkopalveluiden sovelluslokeja [37].

Weblokit jaotellaan yleisesti tapahtumalokiin (engl. transfer log tai access log), virhelokiin (engl. error log), selaintyyppilokiin (engl. agent log) ja viittauslokiin (engl. referrer log). Lokityypeistä eniten käytettyjä ovat tapahtuma- ja virhelokit. [27; 38] Tapahtumalokille kirjataan kaikki WWW-palvelimen prosessoimat pyynnöt, jos pyynnön käsittelyssä tapahtuu virhe, kirjataan tapahtuma myös virhelokille [27]. Tapahtuma-, selaintyyppi- ja viittauslokit ovat tyypillisesti WWW-palvelinohjelmiston vastuulla, kun taas virhelokien kirjaamisesta huolehtivat yksittäiset verkkopalvelut [37].

Dumais ym. määrittelevät julkaisussaan [12] **käyttäytymislokiksi** (engl. behavioral log) lokin, jonne kirjataan ylös ihmisen ja tietojärjestelmän välisiä interaktioita. Käyttäytymislokeja hyödynnetään varsinkin HCI (Human Computer Interaction) -tutkimuksissa, joissa käsitellään ihmisen ja tietokoneen välistä vuorovaikutusta. Keskitettyjen webpohjaisten järjestelmien lisääntyminen on mahdollistanut ihmisten ja webpalveluiden välisten interaktioiden tallentamisen uskomattoman laajassa mittakaavassa. [12]

Weblokeilla on useita käyttötarkoituksia ja lokit ovat käyttökelpoisia vain silloin, kun ne sisältävät riittävästi tietoa käyttötarkoituksiinsa [39]. Webloki sisältää tyypillisesti seuraavat tiedot: tapahtuman aikaleiman, käyttäjän IP-osoitteen, HTTP-pyynnön tyypin, pyydetyn tiedoston URL:in, HTTP-protokollan versionumeron, palvelimen vastauksen tilan, siirrettyjen tavujen määrä, uudelleen ohjatun sivun URL:in ja selaimen tiedot [38;

40]. Jos verkkopalvelu vaatii käyttäjän tunnistamisen, sisältää verkkopalvelun sovellusloki edellisten lisäksi kirjautumisessa käytetyn käyttäjätunnuksen ja tunnistautumisen tilan [37]. Weblokeihin kirjattujen tietojen avulla voidaan selvittää tapahtumia, virheitä, väärinkäyttö- ja tietomurtotilanteita sekä niiden yrityksiä [37]. Tämän lisäksi weblokien tietoja voidaan hyödyntää ihmisen ja tietokoneen välisen vuorovaikutuksen tutkimisessa, käyttäjien käyttäytymisen [12] ja palvelun käytön analysoinnissa [37].

Weblokeja voidaan käyttää tietolähteenä ihmisten toiminnan analysoinnissa. Tähän liittyy kuitenkin paljon erilaisia haasteita, joita käydään tarkemmin läpi seuraavassa kappaleessa.

3.2 Weblokien hyödyntäminen

Weblokien avulla voidaan seurata miljoonia käyttäjiä maailmanlaajuisesti. Tapahtumalokit ja käyttäjätietokannat tarjoavat yhdessä yksityiskohtaista tietoa palveluiden käyttäjistä [28]. Weblokeilla olevan tiedon hyödyntämisessä on kuitenkin ongelmia ja haasteita.

Weblokien sisältämä tieto ei WWW-ympäristön ominaisuuksista johtuen ole aina täysin luotettavaa, vaan tieto voi joltain osin olla puutteellista tai virheellistä. HTTP-protokollan tilattomuus ja välimuistien käyttö verkon eri tasoilla vaikuttavat weblokitiedon luotettavuuteen. Välimuistista haetut sivut eivät luo lokimerkintää alkuperäiselle WWW-palvelimelle, ja POST-metodin kautta välitetty informaatio ei näy weblokeilla [35]. Tämän lisäksi välityspalvelimen kautta tulevat pyynnöt saavat saman tunnisteen, vaikka pyynnöt edustavat eri käyttäjiä. Kuormantasaaja tekniikoita käytettäessä weblokit hajaantuvat usealle eri palvelimelle, jolloin lokitiedot on yhdisteltävä ennen niiden hyödyntämistä. [11]

Vaikka tieto weblokeilla olisi täysin luotettavaa, ei sen avulla pystytä aina muodostamaan totuudenmukaista kuvaa käyttäjän toimista. Analysoinnissa pitää huomioida erinäisiä asioita lokitietojen luonteeseen liittyen [28]. Nicholas ym. mukaan [28] suurin osa weblokeilla olevasta tiedosta pitää kyseenalaistaa ja lokitietoa tulisi käyttää varoen seuraavista syistä:

- Käyttäjät eivät kirjaudu ulos verkosta.
- Käyttäjät voivat olla kirjautuneena verkkoon, mutta eivät käytä sitä.
- Sivun lataaminen ei merkitse sitä, että käyttäjä oikeasti halusi sen.
- Transaktion jäljittäminen yksilöön on lähes mahdotonta.

Kun weblokeja hyödynnetään käyttäjätoimien analysoimiseen, on otettava huomioon yllä kuvatut weblokien ominaispiirteet, tällöin lokitietojen epäluotettavuus ei pääse vaikuttamaan lokitiedoista muodostettuihin analyyseihin [28]. Nicholas ym. toteavat artikkelis-

saan [28], että yksittäisen kuukauden, viikon tai päivän tietoja ei pitäisi tarkastella sellaisenaan, vaan arvo weblokeista löytyy analysoimalla trendejä ja vertailemalla eri ajanhetkiä.

3.3 Henkilötietolaki ja lokien käsittely

Laki asettaa vaatimuksia lokien keräämiselle ja käsittelylle. Vaatimukset koskevat lokien sisältöä, säilytysaikaa, lokitiedon käyttötarkoituksia, sekä lokitietojen eheyden varmistamista. Lokista muodostuu henkilörekisteri, jos lokitiedot sisältävät henkilötietoja. Henkilötiedoksi katsotaan lokitietoon liitettynä mm. nimi- tai sähköpostiosoite -tieto. [39]

Henkilöön liittyvää dataa kerättäessä ja varastoitaessa on selvitettävä henkilörekisteriin liittyvät tietosuojakysymykset. Henkilöille tulee kertoa, mitä tietoa heistä kerätään, mihin tarkoitukseen tietoa kerätään ja miten kerättyä henkilörekisteriä tullaan käyttämään. [26]

Henkilötietolaki määrittää henkilötietojen käsittelylle seuraavat yleiset periaatteet [26]. Huolellisuusvelvoite määrittää rekisterinpitäjän käsittelemään henkilötietoja laillisesti, hyvää tietojenkäsittelytapaa ja huolellisuutta noudattaen. Rekisterinpitäjällä tulee olla toimintansa kautta asialliset perusteet henkilötietojen käsittelyyn ja henkilörekisterin tarkoitus on määriteltävä ennen tietojen keräämistä. Henkilötietoja saa käsitellä vain selkeällä syyllä. Syitä voivat mm. olla rekisteröidyn antama suostumus tai rekisteröidyn asiakas- tai palvelusuhde rekisterinpitäjään. Rekisterinpitäjän on huolehdittava tiedon laadusta; virheellisiä, epätäydellisiä tai vanhentuneita henkilötietoja ei saa käsitellä. Rekisterinpitäjän on laadittava henkilörekisteristä rekisteriseloste, joka on kaikkien saatavilla. Rekisteriselosteesta pitää käydä ilmi rekisterinpitäjän yhteystiedot, henkilötietojen käsittelyn tarkoitus, kuvaus rekisteröityjen ryhmästä ja siihen liittyvistä tiedoista. Rekisteriselosteen tulee kertoa mihin henkilörekisterin tietoja säännöllisesti luovutetaan ja siirretäänkö tietoja EU:n ulkopuolelle. Näiden lisäksi rekisteriselosteen pitää sisältää kuvaus rekisterin suojauksen periaatteista. [5]

Kasvava digitalisaatio ja globalisaatio on tuonut mukanaan uudenlaisia tietosuojakysymyksiä, varsinkin henkilötietojen keräämiseen liittyen. Euroopan Unioni on näihin asioihin liittyen uudistanut tietosuojalakia ja määritellyt uuden tietosuoja-asetuksen, joka tulee voimaan 25.5.2018 kaikissa EU:n jäsenmaissa. Asetuksen tavoitteena on parantaa henkilötietosuojaa ja rekisteröityjen oikeuksia sekä yhtenäistää tietosuojasääntelyä EU-maissa. Uutta tietosuoja-asetusta sovelletaan lähtökohtaisesti kaikkien henkilötietojen käsittelyyn. [3]

4. TIEDONLOUHINTA

Tässä kappaleessa kuvataan tiedonlouhintaan liittyviä määritelmiä ja käsitteitä, sekä kerrotaan mitä tarkoittaa käytön louhinta verkosta. Kappaleessa keskitytään tämän työn kannalta oleellisiin määritelmiin ja käsitteisiin.

4.1 Datasta tiedonlouhintaan

Data on raaka-ainetta, josta informaatio syntyy, data ei itsessään sisällä suhteita tai merkityksiä [36]. Data on numeroita, kirjaimia, kuvia ja ääniä [13; 36]. **Informaatio** on dataa, joka on muutettu merkitykselliseksi kokonaisuudeksi [36]. Dataa voidaan yhdistää kokonaisuuksiksi kontekstin kautta [18]. **Tietämys** on kokoelma informaatiota, jäsentyneitä kokemuksia, arvoja ja oivalluksia. Tietämys tarjoaa viitekehyksen, jonka avulla voidaan arvioida uusia kokemuksia ja uutta informaatiota. [36]

Tiedonlouhinnalla (engl. data mining) tarkoitetaan merkittävän tiedon poimimista suuresta tietoaineistosta [14]. Tiedonlouhinnan avulla tietoaineistoista voidaan löytää liiketoiminnan kannalta hyödyllistä informaatiota [16]. Tiedonlouhinnassa käytetty tietoaineisto on tyypillisesti kerätty jotain muuta tarkoitusta varten. Tiedonlouhinta on poikkitieteellistä; siihen liittyvät mm. tilastotiede, tietokantateknologia, koneoppiminen, hahmontunnistus, tekoäly ja visualisointi. [15]

Tiedonlouhinnan kautta johdettuja yhteyksiä ja yhteenvetoja kutsutaan usein **malleiksi** (engl. model) tai **kaavoiksi** (engl. pattern). Malli on globaali yhteenveto tietoaineistosta, joka ottaa kantaa koko tietoaineistoon. Malli voi esimerkiksi olla rakenne: $Y = aX + c$, jossa Y ja X ovat muuttujia, a ja c ovat mallin parametreja. Mallin parametrien arvot määritellään tiedonlouhinta-prosessin aikana. Kaava on lokaali, eli paikallinen, ja se ottaa kantaa vain rajattuun osaan tietoaineistoa. Esimerkki kaavan rakenteesta voisi olla $Y = aX + c$, kun $X > 0$. Toisin kuin mallit, kaavat kuvaavat rakenteen, joka koskee suhteellisen pientä osaa tietoaineistoa. [15]

Eri tietojen yhdistämistä niin, että yhdistetystä tiedosta saadaan uutta ja oleellista tietoa, kutsutaan **tiedon rikastamiseksi**. Tietoa voidaan rikastaa yhdistelemällä organisaation sisäisiä tietolähteitä tai tuomalla tietoa organisaation ulkopuolelta. [26]

4.2 Kuvaileva ja ennustava tiedonlouhinta

Tiedon analysoinnin näkökulmasta tiedonlouhinta voidaan jakaa kahteen kategoriaan kuvailevaan (engl. descriptive data mining) ja ennustavaan tiedonlouhintaan (engl. predictive data mining). Kuvaileva tiedonlouhinta pyrkii esittämään tietoaineiston yleisiä ominaisuuksia ytimekkäästi summaten. Se kuvaa, miten asiat ovat olleet tai miten ne ovat tällä hetkellä. [14; 24] Ennustava tiedonlouhinta analysoi tietoainestoa muodostaakseen siitä yhden tai useamman mallin, joiden avulla on tarkoitus ennustaa uusien tietoaineistojen käyttäytymistä [14].

Mäntyneva ryhmittelee artikkelissaan [24] kuvailevan tiedonlouhinnat menetelmät seuraavasti: tiivistäminen, visualisointi, klusterointi ja tietojen yhdistäminen ajan suhteen. Tiivistäminen muodostaa tietoaineistosta yhteenvetoja, joiden pohjalta tarkasteltavasta aineistosta voidaan tehdä johtopäätöksiä. Visualisointi pyrkii muuntamaan tietoaineiston visuaaliseen muotoon. Visualisointien avulla tietoaineistosta voidaan tehdä yllättäviä löydöksiä, joita voidaan analysoida tarkemmin tilastollisin menetelmin. Klusterointi pyrkii ryhmittelemään tietoaineiston tarkoituksenmukaisiin ryhmiin tietoalkioiden välisten suhteiden perusteella. Aikasarja-analyysin avulla saadaan lisäymmärrystä siitä, miten muuttujien arvot muuttuvat ajan suhteen. [24; 25]

Käsitteen kuvailu (engl. concept description) on yksinkertaisin kuvailevan tiedonlouhinnan menetelmä. Sen avulla voidaan tehdä ytimekkäitä yhteenvetoja yksittäisistä luokista ja käsitteistä. Käsitteen kuvailuja voidaan johtaa datan karakterisoinnin (engl. data characterization) tai datan erottelun (engl. data discrimination) kautta. Datan karakterisointi muodostaa tiiviin ja ytimekkään yhteenvedon tarkasteltavasta tietoaineistosta. Datan erottelu luo kuvauksia kahden tai useamman tietoaineiston eroista ja samankaltaisuuksista. Käsitteen kuvailu on yhteydessä datan yleistämiseen (engl. data generalization). Käsiteltäessä suuria tietomääriä on käytännöllistä pystyä kuvaamaan käsitteitä yleistetyillä abstraktiotasoilla. Datan yleistäminen mahdollistaa tietoaineiston yleisen käyttäytymisen tutkimisen. [14]

4.3 Tiedonlouhinta verkosta

Tiedonlouhinta verkosta (engl. web mining) tarkoittaa tiedonlouhintamenetelmien soveltamista WWW-ympäristössä. Tiedonlouhinnan tarkoituksena on automaattisesti löytää ja poimia informaatiota verkon dokumenteista ja palveluista [11].

Verkosta tapahtuva tiedonlouhinta voidaan jaotella kolmeen kategoriaan: sisällön louhintaan (engl. web content mining), rakenteen louhintaan (engl. web structure mining) ja käytön louhintaan (engl. web usage mining). [17; 40]

Sisällön louhinta tarkastelee WWW-sivuston sisältödokumenttien rakenteita löytääkseen WWW-sivuston sisällöstä hyödyllistä informaatiota. Rakenteen louhinta pyrkii löytämään WWW-sivuston hyperlinkkien linkkirakenteen ja generoimaan siitä rakenteellisen yhteenvedon. Muodostetun yhteenvedon avulla sivuston rakenteeseen liittyvää dataa voidaan tarkastella. [17] Perehdytään seuraavaksi tässä työssä käytettyyn käytön louhintaan.

4.4 Käytön louhinta verkosta

Käytön louhinta verkosta pyrkii löytämään kiinnostavaa tietoa verkon sekundaarisesta datasta. Tietoaineistona käytön louhinnassa voivat olla WWW-palvelimen ja selainten lokitiedostot, käyttäjäprofiilit sekä kirjanmerkit. [11; 17] Käytön louhinnan tavoitteena on löytää tietoaineistosta verkon käyttöön liittyviä kaavoja [35].

Käytön louhinnan kautta saatua informaatiota voidaan organisaatiossa hyödyntää usealla tavalla. Informaatio WWW-sivuston ladatuimmista sivuista, ahkerimmista käyttäjistä ja toistuvista käyttöajankohdista voivat auttaa löytämään potentiaalisia asiakkaita, käyttäjiä ja markkinoita [14]. Louhitun informaation avulla voidaan parantaa WWW-sivuston toimintakykyä ja turvallisuutta, sekä tarjota tukea organisaation markkinointipäätöksiin [35].

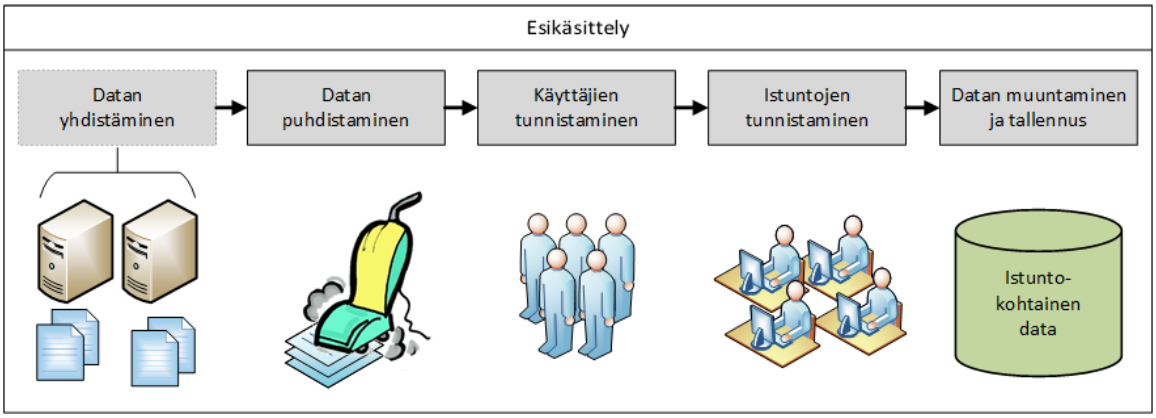
Käytön louhinta -prosessi voidaan jakaa kolmeen vaiheeseen: esikäsittelyyn (engl. preprocessing), kaavojen löytämiseen (engl. pattern discovery) ja kaavojen analysoimiseen (engl. pattern analysis). [17; 27; 38] Seuraavissa kappaleissa tarkastellaan käytön louhinta -prosessin eri vaiheita tarkemmin.

4.4.1 Datan esikäsittely

WWW-ympäristön informaatio on heterogeenistä ja strukturoimatonta, minkä vuoksi datan esikäsittely on keskeinen vaihe käytön louhinnassa verkosta [17]. Usein raaka lokidata pitää siivota, tiivistää ja muuntaa, jotta siitä pystytään poimimaan ja analysoimaan merkittävää ja hyödyllistä informaatiota [14].

Kappaleessa 3.2 luetellut haasteet weblokien hyödyntämisessä pitää huomioida weblokien esikäsittelyssä. Lisäksi pitää varmistua siitä, että kaikki tarpeelliset tapahtumat tallentuvat weblokeille. Käytön louhinta -prosessissa kerätty informaatio on hyödyllistä vain, jos weblokidata antaa paikkansapitävän kuvan tapahtumista. [9]

Esikäsittelyssä on useita vaiheita (Kuva 2): datan puhdistaminen, käyttäjien ja istuntojen tunnistus, sekä lokitiedon muuntaminen muotoon josta sitä voidaan seuraavassa vaiheessa hyödyntää [11; 17; 27]. Jos käytettävä tietoaineisto sijaitsee useassa lähteissä, on eri lähteiden data yhdistettävä ennen datan puhdistusta [11].



Kuva 2. *Datan esikäsittelyn vaiheet*

Datan puhdistuksessa lokidatasta poistetaan epäolennaiset pyynnöt ja kentät. Epäolennaisia pyyntöjä ovat latauspyynnöt HTML-sivun lisäresursseihin (esim. kuva- tai skriptitiedostot), sekä erilaisten robottiohjelmistojen muodostamat pyynnöt. [11; 17] Jafari ym. [17] lisäävät epäolennaisiin pyyntöihin myös kaikki virheeseen päätyneet pyynnöt.

Käyttäjien tunnistus -vaiheessa yksittäisten käyttäjien lokimerkinnät erotellaan toisistaan IP-osoitteeseen, käyttöjärjestelmä- ja selaintietoihin perustuvien heuristiikkojen avulla [11; 17]. Istuntojen tunnistus -vaiheessa yksittäisen käyttäjän suorittamat toiminnot ryhmitellään yhteen, minkä jälkeen samaan vierailuun kuuluvat toiminnot jaotellaan edelleen omiin ryhmiinsä [19]. Monia erilaisia istuntojen tunnistusmenetelmiä on olemassa, menetelmät jaotellaan usein aikaperusteisiin ja osoitusperusteisiin (engl. referrer-based) [19]. Käyttäjien ja istuntojen tunnistaminen lokitiedosta yksinkertaistuu, jos sivustolle pitää kirjautua tai jos sivustolla käytetään evästeitä. Tällöin tarvittavat tiedot käyttäjän ja istunnon tunnistukseen löytyvät weblokeilta. [11]

Viimeisessä esikäsittelyn vaiheessa lokitiedoista valitaan kentät, joita halutaan tarkastella [11] ja lokitieto muunnetaan muotoon, jota voidaan helposti seuraavassa vaiheessa käsitellä [17; 27].

4.4.2 Kaavojen löytäminen

Kaavojen löytämisvaiheessa esikäsitellystä datasta poimitaan käyttäjien käyttäytymiseen liittyviä kaavoja [17]. Kaavojen löytämiseen on olemassa useita erilaisia menetelmiä. Käytön louhinnan yhteydessä yleisesti käytettyjä menetelmiä ovat tilastollinen analyysi, assosiaatiosäännöt, klusterointi, luokittelu, sekvenssikaavat ja riippuvuusmallinnus. Näistä tilastollinen analyysi on yleisin menetelmä, jonka avulla tietoa poimitaan WWW-sivuston käyttäjistä. [17; 35]. Tässä työssä hyödynnetään tilastollista analyysiä kaavojen löytämiseen ja tietojen poimintaa.

Kaavojen löytämisvaiheessa aiemmin muodostettuja istuntotietoja voidaan analysoida erilaisten kuvailevien tilastollisten analyysien avulla [35]. Kuvaileva tilastotiede keskittyy menetelmiin, joilla tarkasteltavaa ainestoa voidaan havainnollisesti ja tehokkaasti esitellä [22]. Aineiston kuvailu tapahtuu aineistosta laskettavien tunnuslukujen ja visualisointien avulla [22]. Kuvaileva tilastotiede auttaa ymmärtämään tietoaineiston jakaumaa. Tiedonlouhinnassa halutaan usein karakterisoida dataa, sekä datan keskittyneisyyden että hajonnan osalta. Keskittyneisyyden tunnuslukuja ovat keskiarvo, mediaani, moodi ja vaihteluvälin keskipiste. Hajonnan tunnuslukuja ovat kvartaalit, poikkeavat havainnot ja varianssi. [14]

Jafari ym. [17] jaottelevat weblokeilta saatavan tilastollisen tiedon aktiivisuus-, vianmääritys- ja palvelintilastoihin. Sivuston aktiivisuutta kuvaavat tilastot sisältävät tietoja: kävijämääristä, eri pyyntöjen (onnistuneet, epäonnistuneet, uudelleenohjatut) lukumääristä, keskimääräisistä sivujen katseluajoista ja sivuston keskimääräisestä navigointipolun pituudesta. Vianmääritystilastoista löytyvät virhetilanteisiin ja niiden diagnosointiin liittyvät tiedot. Palvelintilastot keräävät tietoa palvelimen toiminnasta ja niiltä voidaan nähdä mm. vierailluimmat sivut, sekä käytetyimmät sisääntulo- ja poistumissivut. [17]

Vaikka tilastolliset menetelmät suorittavat vain summittaisen analyysin esikäsitellystä datasta, voi menetelmien avulla hankittu tietämys olla hyödyllistä [17].

4.4.3 Kaavojen analysointi

Kaavojen analysoinnissa (engl. pattern analysis) on kaksi keskeistä tavoitetta: poimia kiinnostavat tilastot, säännöt tai kaavat edellisen vaiheen tuloksista [17; 27; 35] ja hankkia informaatiota, joka voi tarjota arvokaita oivalluksia sivuston käyttäjien käyttäytymisestä [17]. Han ym. kuvaavat kirjassaan [14] kiinnostavan kaavan edustavan tietämystä. Kiinnostava kaava on ihmisten ymmärrettävissä, voimassa oleva, hyödyllinen ja ennennäkemätön. Kaavan kiinnostavuudelle on olemassa useita objektiivisia mittoja, jotka perustuvat löydetyn kaavan rakenteeseen ja kaavan lähtökohtana olevaan tilastotieteeseen. Objektiiviset mitat ovat kuitenkin yksinään riittämättömiä määrittämään kaavan kiinnostavuutta; sen sijaan tarvitaan subjektiivisia mittoja, jotka kuvaavat spesifisen käyttäjän tarpeita ja kiinnostuksen kohteita. [14]

Käyttäytymiseen liittyvien kaavojen löytäminen erilaisten menetelmien avulla ei olisi hyödyllistä, ellei olisi olemassa mekanismeja ja työkaluja, joiden avulla analyytikot pystyvät paremmin ymmärtämään niitä. Tarvitaan työkaluja ja menetelmiä, joilla löydettyjä kaavoja voidaan analysoida. Kuten kaavojen löytämiseen myös kaavojen analysointiin käytetyt menetelmät ovat peräisin usealta eri tieteenalalta: tilastotieteestä, grafiikasta ja visualisoinnista, käytettävyys analyyseistä ja tietokantakyselyistä. [9]

Kaavojen analysointiin valitun analysointimenetelmän määrittää sovellus, jolle tiedonlouhinta toteutetaan [17; 35]. Käytön louhinnan yhteydessä käytettyjä kaavojen analysointimenetelmiä ovat tietämyskyselyt (engl. knowledge query), OLAP-operaatiot ja visualisointi [9; 35].

Tietämyskyselyt kohdistuvat tiedonlouhintaprosessissa poimittuun tietämykseen [9]. Tietämyskyselyiden, kuten SQL:n, avulla löydettyjä kaavoja voidaan analysoida ja rajata erilaisten ehtojen kautta [9; 17; 35]. OLAP-operaatioita varten sivuston käyttöä koskeva data ladataan datakuutioon [35]. Datakuutiossa data organisoidaan useisiin dimensioihin, joista jokainen sisältää useita käsitehierarkian muodostamia abstraktiotasoja. Nämä dimensiot mahdollistavat datan tarkastelun useasta eri perspektiivistä [14]. Visualisointia on käytetty onnistuneesti auttamaan ihmisiä ymmärtämään erilaisia ilmiöitä [9]. Visualisointitekniikoiden avulla voidaan korostaa kokonaisvaltaisia kaavoja tai trendejä tietoaineistossa [17; 35]. Löydettyjen kaavojen visualisointi helpottaa käyttäjiä identifioimaan kiinnostavat kaavat [14].

5. JÄRJESTELMÄN TEKNINEN TOTEUTUS

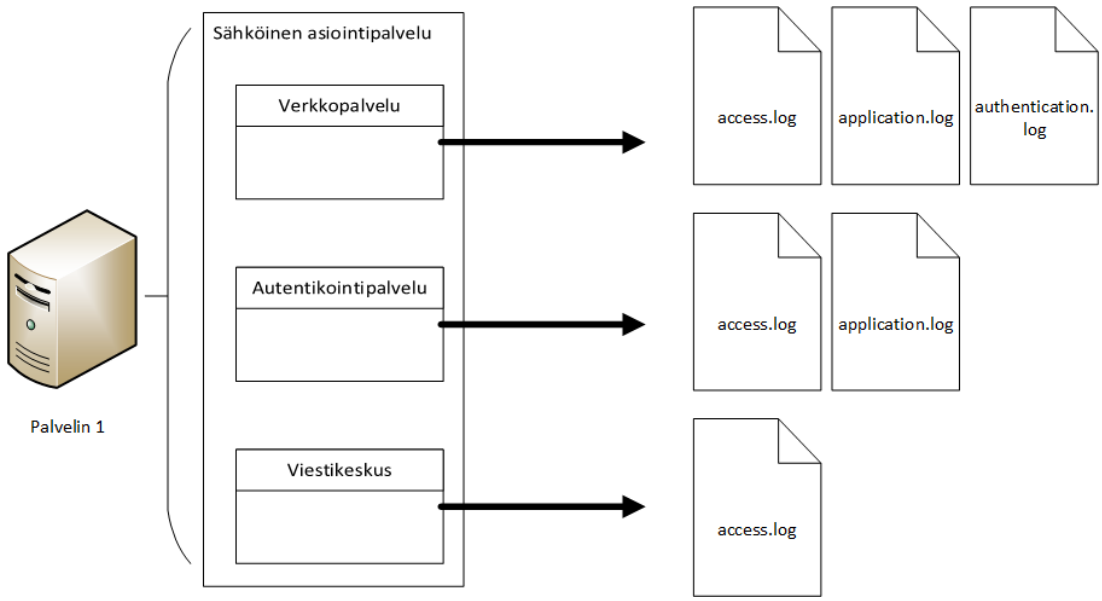
Tutkimuskysymystä lähdettiin ratkaisemaan toteuttamalla tilaajaorganisaatiolle järjestelmä, jonka avulla sähköisen asiointipalvelun lokitiedostoista tuotetaan informaatiota päätöksenteon tueksi. Tässä luvussa kuvataan yleisellä tasolla sähköisen asiointipalvelun arkkitehtuuri, työssä käytetyt teknologiat ja työn tekninen toteutus.

5.1 Sähköinen asiointipalvelu

Tilaajaorganisaation sähköinen asiointipalvelu koostuu kolmesta erillisestä sovelluksesta: verkkopalvelusta, autentikointipalvelusta ja viestikeskuksesta. Verkkopalvelu sisältää sähköisen asioinnin eri palvelut. Se hyödyntää palveluiden toteutuksessa autentikointipalvelua ja viestikeskusta. Autentikointipalvelu huolehtii käyttäjien autentikoinnista ja käyttäjäistuntojen oikeellisuudesta. Viestikeskus puolestaan mahdollistaa palvelun käyttäjien ja organisaation välisen kommunikaation. Tässä työssä keskityttiin louhimaan tietoa verkko- ja autentikointipalvelun sovelluslokeista.

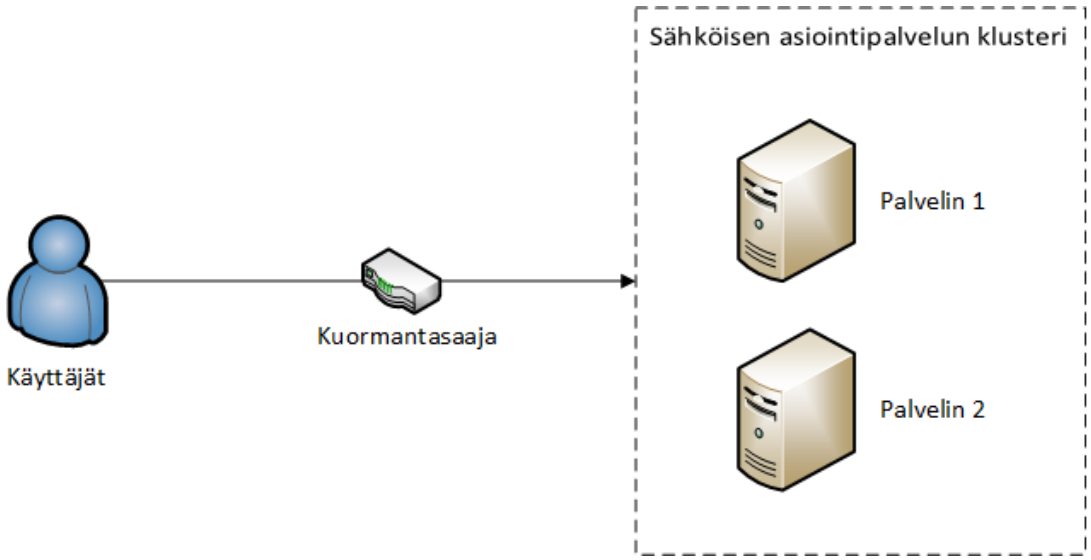
Verkkopalvelusovellus on toteutettu yhden sivun arkkitehtuurilla (engl. Single-Page Application). Arkkitehtuurista johtuen käyttäjien toimia voidaan seurata palvelimella sijaitsevista lokeista vain karkealla tasolla, sillä käyttäjän toimet eivät aina synnytä palvelimelle välittyvää pyyntöä. Autentikointipalvelu on arkkitehtuuriltaan perinteinen websovellus, jossa käyttäjän toimet välittyvät palvelimelle. Molemmat sähköisen asiointipalvelun sovellukset koostuvat edustasta (engl. frontend) ja taustapalvelusta (engl. backend).

Sähköisen asiointipalvelun sovellukset kirjaavat tapahtumiaan useaan eri lokitiedostoon, kts. Kuva 3. *Access.log* -tiedostoihin kirjautuvat käyttäjän palvelussa tekemät toimenpiteet, josta on muodostunut palvelimelle välitettävä pyyntö. Tarkempaa teknistä tietoa sovellusten toiminnasta kirjataan *application.log* -tiedostoihin. *Authentication.log* -tiedosto sisältää tietoa käyttäjien rooleista ja autentikaatiosta.



Kuva 3. *Sähköisen asiointipalvelun lokitiedostot*

Organisaation sähköinen asiointipalvelu on hajautettu usealle eri palvelimelle. Sähköisen asiointipalvelun käyttäjä ohjataan kuormantasaajan avulla yhdelle palvelimista (Kuva 4).



Kuva 4. *Sähköisen asiointipalvelun palvelimet*

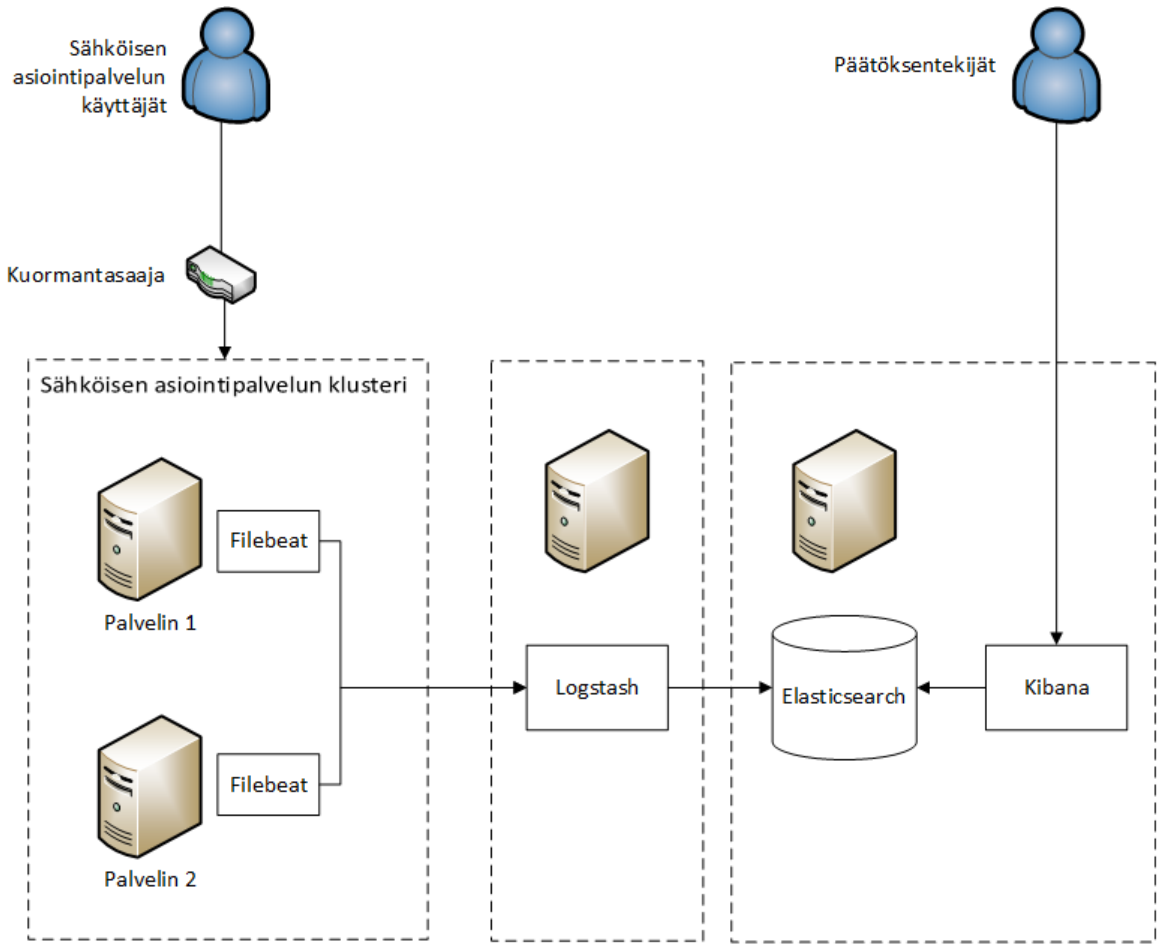
Tilaajaorganisaation sähköinen asiointipalvelu on tarkoitettu vain organisaation asiakkaille. Ottaessaan sähköisen asiointipalvelun käyttöönsä asiakas hyväksyy palvelun käyttöehdot, joissa kerrotaan mitä tietoja palvelun käytöstä kerätään ja mihin tietoja käytetään. Sähköisestä asiointipalvelusta löytyy linkki rekisteri- ja tietosuojaselosteeseen. Henkilötietolaki ja uusi EU:n tietoturva-asetus on työssä huomioitu anonymisoimalla

data niiltä osin, jonka avulla käyttäjä voitaisiin ilman ulkopuolista dataa tunnistaa. Palvelun sovelluslokeista ei siirretä Elasticsearch:iin asiakkaiden nimiä, osoitteita, sähköpostiosoitteita ym. yksilöiviä tietoa.

5.2 Elastic-ohjelmistopino

Lokitietojen analysointiin on olemassa useita erilaisia sovelluksia ja työkaluja. Osa työkaluista on maksullisia ja osa ilmaisia avoimen lähdekoodin ohjelmistoja. Työn toteutusteknologioita valittaessa pohdittiin teknologioiden soveltuvuutta ongelman ratkaisemiseen, järjestelmän ylläpidettävyyttä, käyttöönoton helppoutta sekä projektin kustannuksia. Organisaation informaation tarve keskittyi nykytilanteen kuvailuun, joka ei vaadi monimutkaista tiedon prosessointia. Tämän lisäksi käsiteltävien lokien tietomäärät olivat suhteellisen pieniä. Organisaatiossa oli käytössä keskitetty lokienhallintajärjestelmä, joka oli toteutettu Elastic-ohjelmistopinon avulla. Näistä syistä työssä päädyttiin käyttämään organisaatiossa jo käytössä ollutta Elastic-ohjelmistopinoa.

Elastic-ohjelmistopinoa ylläpitää ja kehittää Elastic -ohjelmistoyritys. Elastic-ohjelmistopino mahdollistaa laajan räätälöinnin, erilaisen data viennin järjestelmään ja tiedonsiirron edelleen toisiin järjestelmiin. Elastic-ohjelmistopinoa on helppo skaalata tietomäärien kasvaessa ja se on helppo integroida osaksi muita järjestelmiä. Elastic-ohjelmistopino on avoimen lähdekoodin ohjelmisto ja se koostuu Elasticsearch, Logstash ja Kibana -ohjelmistoista. Lokitiedostojen välittämiseen sähköisen asioinnin klusterista eteenpäin käytettiin Filebeat-ohjelmaa (Kuva 5).



Kuva 5. *Sähköinen asiointipalvelu ja Elastic-ohjelmistopino*

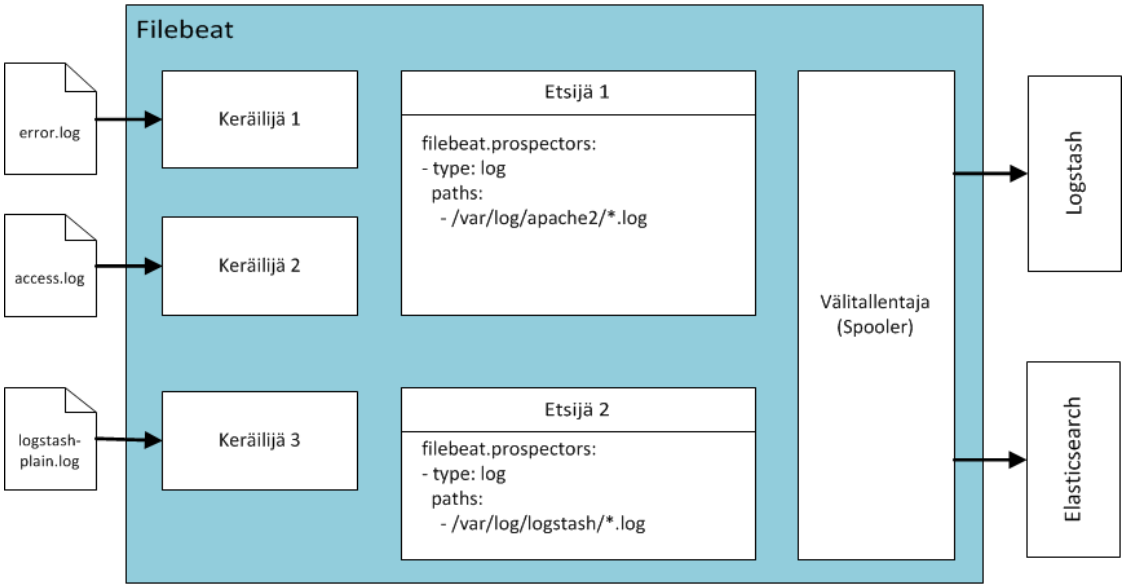
5.2.1 Filebeat

Filebeat kuuluu Beats-ohjelmiin. Beats-ohjelmat ovat kevyitä agenttiohjelmia, joita voidaan asentaa palvelimiin lähettämään operatiivista dataa eteenpäin. Beats-ohjelmat on kirjoitettu Go-ohjelmointikielellä ja ne käyttävät *libbeat*-kirjastoa, joka tarjoaa rajapinnan tiedon lähettämiselle. Beats-ohjelmia on useita eri käyttötarkoituksiin, joista Elastic tukee virallisesti seuraavia: Filebeat, Heartbeat, Metricbeat, Packetbeat ja Winlogbeat. Näiden ohjelmien lisäksi avoimen lähdekoodin kehittäjäyhteisö on luonut useita muita Beats-ohjelmia. [1]

Filebeat on tiedonvälittäjä, jonka avulla paikallisten lokitiedostojen tiedot voidaan välittää eteenpäin Elastisearch:lle tai Logstash:lle. Filebeat koostuu kahdesta pääkomponentista etsijästä (engl. prospector) ja keräilijästä (engl. harvester). [4]

Keräilijät lukevat yksittäisen tiedoston sisältöä rivi kerrallaan ja lähettävät sisällön eteenpäin ulostulolle. Jokaiselle tiedostolle on oma keräilijänsä, joka on vastuussa tiedoston avaamisesta ja sulkemisesta. Etsijät hallinnoivat keräilijöitä ja vastaavat siitä, että kaikki

konfiguraatiotiedostossa määritetyt lähdetiedostot löytyvät. Etsijä käy läpi jokaisen tiedoston ja tarkistaa voidaanko tiedosto ohittaa vai tarvitseeko tiedostolle käynnistää keräilijä tai onko keräilijä jo käynnistetty (Kuva 6). [4]



Kuva 6. *Filebeat: etsijät ja keräilijät*

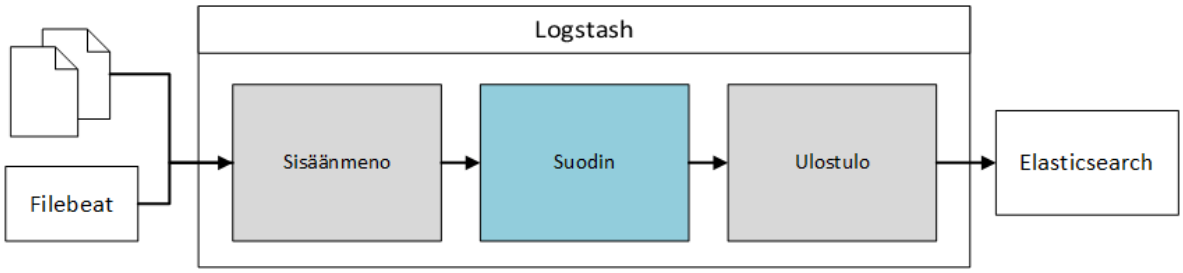
Filebeat pitää kirjaa jokaisesta tiedostosta ja lähetyksestä, tiedot tallennetaan säännöllisesti rekisteritiedostoon. Kun Filebeat käynnistyy, rekisteritiedoston dataa käytetään rakentamaan järjestelmän aiempi tila. Jos määrätty ulostulo ei ole tavoitettavissa, Filebeat kirjaa ylös viimeiset lähetetyt rivit ja jatkaa tiedoston lukemista vasta kun ulostulo on taas tavoitettavissa. Filebeat huolehtii myös, että jokainen tiedoston rivi lähetetään ainakin kerran ulostulolle seuraamalla lähetyksen tilaa. Filebeat yrittää lähettää tietoa uudelleen niin kauan, kunnes ulostulo kuittaa tiedon vastaanotetuksi. Näiden toimenpiteiden avulla Filebeat varmistaa, että kaikki tiedoston rivit tulevat lähetetyiksi. [4]

5.2.2 Logstash

Logstash-ohjelmisto on tiedonkerääjä, jonka avulla sekä strukturoitua että strukturoimatonta dataa voidaan kerätä, jäsennellä ja analysoida. Logstash pystyy dynaamisesti yhdistämään dataa eri lähteistä ja normalisoimaan sen eri ulostuloja varten. Logstash voidaan yhdistää eri tietolähteisiin ja ulostuloihin lisäosien avulla. Tiedon jäsentely tapahtuu Logstash:ssa erilaisten lisäosien avulla. [7; 10]

Logstash on toteutettu JRuby:lla ja se noudattaa liukuhihna-arkkitehtuuria (engl. pipeline), jossa on kolme vaihetta: sisäänmeno (engl. inputs), suodin (engl. filters) ja ulostulo (engl. outputs). Sisäänmeno luo tapahtumia vastaanottamastaan tiedosta, suodin muokkaa niitä ja ulostulo lähettää ne eteenpäin (Kuva 7). Logstash:n konfigurointitiedostossa on oma lohko jokaiselle liukuhihnan vaiheelle. Lohkoissa määritellään vaiheessa käytetyt

lisäosat ja niiden asetukset. Konfigurointitiedoston lohkot voidaan eriyttää myös omiksi tiedostoikseen. [7]



Kuva 7. *Logstash: liukuhihna-arkkitehtuuri*

Tässä työssä Logstash määriteltiin vastaanottamaan Filebeat-ohjelmistojen lähettämiä tietoja (Kuva 5). Logstash luo sisäänmenovaiheessa vastaanotetusta tiedosta tapahtuman ja vie vastaanotetun lokirivin sisällön oletuksena tapahtuman *message*-kenttään [7]. Suodinvaiheessa vastaanotettu tapahtuma jäsennellään haluttuun muotoon ja tapahtuman tietoja rikastetaan jdbc_streaming-lisäosaa hyödyntämällä. Tapahtuman jäsentely toteutettiin grok, mutate, kv, ruby, date ja json -lisäosien avulla. Suodinvaiheessa hyödynnettiin myös lisäosien yhteisiä ominaisuuksia, joita kaikki lisäosat tukevat [7]. Yhteisiin ominaisuuksiin kuuluvat mm. kentän tai tunnisteen poistaminen tapahtumasta ja kentän tai tunnisteen lisääminen tapahtumaan. Jäsentelyn aikana lisäosat voivat määritellä tapahtumalle erilaisia tunnisteita. Tunnisteet lisätään tapahtuman *tags*-kenttään ja ne ovat usein merkki jäsentelyn epäonnistumisesta. Lopuksi jäsennelty tapahtuma välitetään Logstash:n ulostulovaiheessa eteenpäin Elastisearch:lle (Kuva 5).

Logstash:n suodinvaiheessa tiedon jäsentelyyn käytetyt lisäosat löytyvät esimerkkeinä Kuvassa 8. Lähteenä esimerkin lisäosien toiminnan kuvaamisessa on käytetty Logstash-ohjelmiston virallista referenssiä [7]. Grok-lisäosa jäsentelee ja strukturoi satunnaista tekstiä. Lisäosa vertailee vastaanotettua tapahtumaa kaavoihin ja jäsentelee tapahtuman tiedot kaavoja vastaaviin kenttiin. Grok-lisäosassa käytetyt kaavat ovat muotoa *%{SYNTAKSI:merkitys}* ja perustuvat säännöllisiin lausekkeisiin (engl. regular expression). Logstash-ohjelmistossa on oletuksena 120 valmiiksi määriteltyä grok-kaavaa, joita voidaan yhdistellä. Tämän lisäksi omia grok-kaavoja voidaan lisätä ohjelmistoon tarpeen mukaan. Grok-lisäosa on määritelty esimerkissä (Kuva 8) riveillä 2-7. Rivit 4 ja 5 määrittelevät grok-kaavat joihin *message*-kentän dataa yritetään sovittaa. Kun grok-kaavoja on määritelty useita, käydään ne järjestyksessä läpi, kunnes sopiva kaava löytyy. Jos sopivaa kaavaa ei löydy tapahtumaan lisätään *_grokparsefailure*-tunniste.

```
1  filter {
2      grok{
3          match => [
4              "message", "%{TIME:log_time} %{WORD:HTTP_method} %{URIPATH:url}%{URIPARAM:params}",
5              "message", "%{TIME:log_time} %{WORD:method}\(%{DATA:data}\)"
6          ]
7          remove_field => [ "message" ]
8      }
9
10     if [params]{
11         kv {
12            source => "params"
13            field_split => "&?"
14            include_keys => [ "param1", "param3" ]
15         }
16
17         ruby {
18           code => "
19                  new_value = event.get('[param1]')[0..1]
20                  event.set('language_code',new_value)
21                  "
22         }
23
24         mutate {
25             convert => { "param3" => "integer" }
26             add_field => {"timestamp" =>"%{+YYYY-MM-dd}_%{log_time}"}
27             remove_field => [ "params", "log_time" ]
28         }
29     }
30
31     if [data]{
32         json {
33             source => "data"
34             target => "json_data"
35             add_field => {"timestamp" =>"%{+YYYY-MM-dd}_%{log_time}"}
36             remove_field => [ "data", "log_time" ]
37         }
38     }
39
40     date{
41         match => [ "timestamp", "YYYY-MM-dd_HH:mm:ss.SSS", "ISO8601"]
42         timezone => "Europe/Helsinki"
43         remove_field => [ "timestamp" ]
44     }
45 }
```

Kuva 8. *Esimerkki suodinvaiheessa käytetyistä lisäosista*

Kuvitellaan että esimerkin (Kuva 8) suodinosalle annetaan jäsenneltäväksi lokirivi ”*08:10:18.200 GET /auth/success?param1=FI2018&param2=value2&param3=300”*. Tällöin grok-lisäosan rivillä 4 kuvattu grok-kaava jäsentelee lokirivin seuraavasti:

- log_time: 08:10:18.200
- HTTP_method: GET
- url: /auth/success
- params: ?param1=FI2018&param2=value2&param3=300

Kun vastaanotettu tapahtuma on jäsennelty kenttiin, voidaan kenttiä jäsennellä ja muokata edelleen lisäosien avulla. Tapahtuman jäsentely on usein ehdollistettua ja riippuu tapahtuman ominaispiirteistä. Esimerkissä on kaksi ehtoa (Kuva 8: rivi 10 ja 32) jotka tarkistavat että tapahtumasta löytyy tarvittava kenttä ennen ko. kentän jäsentelyä. Tarkasteltavasta tapahtumasta löytyy *params*-kenttää, joten jäsentely jatkuu esimerkin riviltä 11.

Kv-lisäosa jäsentelee automaattisesti kenttiä, jotka sisältävät *”avain=arvo”*-muotoista dataa. Kuvan 8 esimerkissä kv-lisäosa on määrittely riveillä 11-15. Arvoparien erotinmerkit määritellään *field_split*-kentässä ja *include_keys*-asetus määrittää mitkä jäsennellyistä arvopareista lisätään tapahtumaan. Kv-lisäosa jäsentelee lokirivin *params*-kentän arvot ja lisää tapahtumaan kentät *param1* ja *param3*.

Ruby-lisäosa mahdollistaa ruby-koodin hyödyntämisen tapahtumien jäsentelyssä. Suoritettava koodi voidaan määritellä lisäosassa tai koodi voi sijaita erillisessä tiedostossa. Aikaisemmin kuvatussa esimerkissä (Kuva 8: rivit 17-22) ryby-koodi on määritelty lisäosan *code*-kentässä. Koodi poimii *param1*-kentän kaksi ensimmäistä merkkiä ja tallentaa ne uuteen *language_code*-kenttään. Ryby ja kv -lisäosien jälkeen tapahtuma on jäsennelty muotoon:

- log_time: 08:10:18.200
- HTTP_method: GET
- url: /auth/success
- params: ?param1=FI2018&param2=value2&param3=300
- param1:FI2018
- param3:300
- language_code:FI

Mutate-lisäosa mahdollistaa tapahtuman kenttien monipuolisen muokkaamisen, kenttiä voidaan mm. nimetä uudelleen, korvata, yhdistellä, pilkkoa ja konvertoida. Kuvan 8 esimerkissä mutate on määritelty riveillä 24-28. Esimerkin rivillä 25 *param3*-kentän arvo konvertoidaan kokonaisluvuksi. Riveillä 26-27 hyödynnetään lisäosien yhteisiä ominaisuuksia. Rivi 26 lisää tapahtumaan *timestamp*-kentän, kentän sisältö muodostetaan yhdistelemällä kuluvapäivä (muodossa *YYYY-MM-dd*) ja *log_time*-kentän arvo. Rivi 27 poistaa kentät *params* ja *log_time*. Mutate-lisäosan jälkeen esimerkkinä käytetty lokirivi on jäsennelty kenttiin seuraavasti:

- HTTP_method: GET
- url: /auth/success
- param1:FI2018
- param3:300
- language_code:FI
- timestamp: 2018-04-01_08:10:18.200

Tapahtuman jäsentelyssä tulee seuraavaksi vastaan esimerkin toinen ehto (Kuva 8: rivi 31). Koska tarkasteltavasta tapahtumasta ei löydy *data*-nimistä kenttää, jatkuu tapahtuman jäsentely riviltä 41.

Date-lisäosaa käytetään päivämäärien ja aikaleimojen jäsentelyyn. Kuvan 8 esimerkissä date-lisäosa on määritelty riveillä 41-46, lisäosa jäsentelee *timestamp*-kentän sisällön ai-

kaleimaksi, joka on muotoa *YYYY-MM-dd_HH:mm:ss.SSS*. Aikaleiman jäsentelyssä käytetty aikavyöhyke määritellään lisäosan *timezone*-kentässä. Jos date-lisäosassa ei ole määritelty kenttää, johon aikaleima tallennetaan, tallennetaan se tapahtuman aikaleimaksi, *@timestamp*-nimiseen kenttään. Päivämäärän tai aikaleiman jäsentelyn epäonnistuessa tapahtumaan lisätään *_dateparsefailure*-tunniste. Esimerkkinä käytetty lokirivi olisi lopulta jäsennelty muotoon:

- HTTP_method: GET
- url: /auth/success
- param1:FI2018
- param3:300
- language_code:FI
- @timestamp: 2018-04-01’T’08:10:18.200’Z’

Suodinvaiheen json-lisäosa mahdollistaa JSON-muotoisen datan jäsentelyn. Lisäosa jäsentelee kentän sisältämän JSON-datan ja lisää datan tapahtumaan. Jäsentelyn epäonnistuessa tapahtumaan lisätään tunniste *_jsonparsefailure*. Jos esimerkin (Kuva 8) suodinosalle annetaan jäsenneltäväksi lokirivi ”*08:14:20.100 UpdateForm({"id":"lomake_A", "field1":"value1", "field2": "value2"})”,* jäsentelee grok-lisäosa ensin tapahtuman kenttiin rivillä 5 olevan kaavan mukaisesti:

- log_time: 08:14:20.100
- method: UpdateForm
- data: {"id":"lomake_A", "field1":"value1", "field2": "value2"}

Tämän jälkeen json-lisäosa (Kuva 8: rivit 32-37) jäsentelee *data*-kentässä olevan JSON-muotoisen datan *json_data*-nimiseen kenttään. Tämän lisäksi json-lisäosa lisää tapahtumaan *timestamp*-kentän, sekä poistaa tapahtumasta *data* ja *log_time* -kentät, samaan tapaan kuin aiemmin mutate-lisäosan yhteydessä. Json-lisäosan jälkeen esimerkkitapahtuma olisi muotoa:

- method: UpdateForm
- json_data.id: lomake_A
- json_data.field1: value1
- json_data.field2: value2
- timestamp: 2018-04-01_08:10:18.200

Lopuksi date-lisäosa lisää tapahtumaan *@timestamp*-kentän samoin kuin ensimmäisen esimerkkitapahtuman kohdalla.

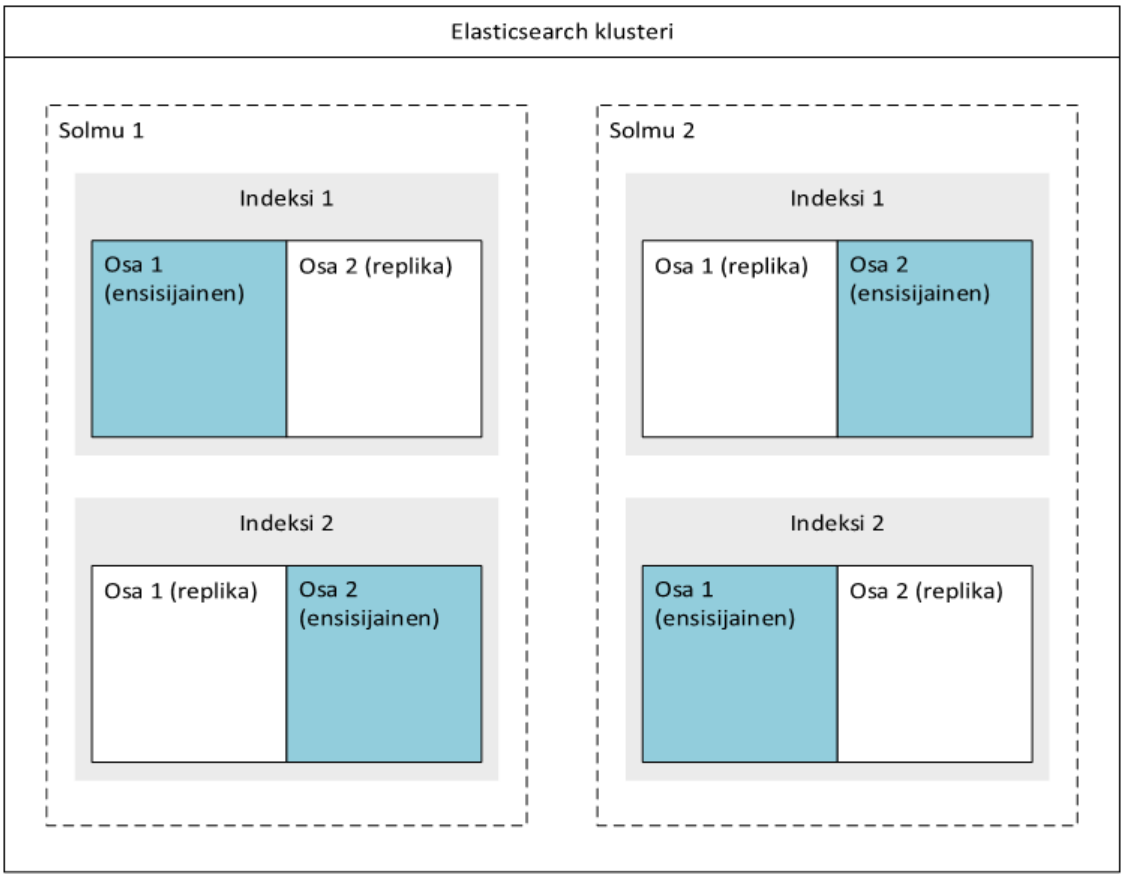
5.2.3 Elasticsearch ja Kibana

Elasticsearch on hajautettu tekstihaku- ja analytiikkamoottori. Elasticsearch perustuu Apachen Lucene-hakukonekirjastoon. Ensimmäinen versio Elaticsearch:sta julkistettiin vuonna 2010. Uusimpien versioiden myötä Elasticsearch:n käyttöä on laajennettu myös tiedon taltiointiin NoSQL -tietovarastona. Elasticsearch mahdollistaa isojen tietomäärien tallennuksen, haun ja analysoinnin lähes reaaliajassa. [10]

Elasticsearch on toteutettu Javalla, ja se noudattaa REST-suunnittelumallia, jossa kaikki CRUD (Create, Read, Update, Delete) -toiminnot on toteutettu käyttämällä REST-päätepisteitä. Usealle ohjelmointikielelle on olemassa valmiit asiakaskirjastot, joiden avulla Elasticsearch:ia voidaan käyttää REST-rajapinnan kautta. Näistä syistä Elasticsearch:n integroiminen muihin järjestelmiin on helppoa. [10]

Elasticsearch-klusteri koostuu yhdestä tai useammasta solmusta (engl. node). Solmu on yksittäinen Elasticsearch -instanssi (Kuva 9). Klusteri säiliöi kaiken datan sekä tarjoaa yhteiset indeksointi- ja hakutoiminnot kaikkien klusterin solmujen läpi. Elasticsearch tallentaa datan JSON-muotoisina dokumentteina. Dokumentit tallennetaan indekseihin ja saman indeksin dokumenteilla on samankaltaiset ominaisuudet. Hajautettavuuden aikaansaamiseksi indeksit voidaan jakaa useampaan osaan (engl. shard), jotka voivat sijaita yhdellä tai useammalla solmulla. Yksittäisiä indeksin osia voidaan replikoida saatavuuden ja vikasietoisuuden parantamiseksi. [2; 10]

Jokaisella indeksillä on kuvaus (engl. mapping), joka sisältää tiedon siitä, miten dokumenttien kentät indeksoidaan ja tallennetaan Elasticsearch:ssa [10]. Elasticsearch:n dynaaminen kuvaus (engl. dynamic mapping) luo ja täydentää indeksin kuvausta automaattisesti, kun indeksiin lisätään dokumentteja [2].



Kuva 9. *Elasticsearch-klusteri, jossa indeksit on ositettu kahteen osaan ja replikoitu kertaalleen*

Tässä työssä Elasticseach:a käytetään tietovarastona, jonne Filebeat-ohjelmien lähettämät lokitiedon viedään Logstash-ohjelmiston jäsenneltyä ne (Kuva 5). Tiedonlouhinnassa hyödynnetään Elasticserch:n kyselyitä ja aggregointeja sekä mahdollisuutta hyödyntää skriptejä erilaisen operaatioiden suorittamiseen.

Elasticsearch:ssa kyselyitä suoritetaan query DSL(Domain Specific Language) -kyselykielen avulla. Query DSL hyödyntää Elasticsearch:n tarjoamaa JSON-rajapintaa, jonka avulla kyselyitä voidaan kirjoittaa JSON-muodossa. Kyselyt koostuvat kahden tyyppisistä lauseista: lehti- (engl. leaf query clause) ja koostekyselylauseista (engl. compound query clause). Lehtikyselylause etsii määrättyä arvoa määrätystä dokumentin kentästä ja koostekyselylause yhdistää yksittäisiä kyselylauseita. [2; 10]

Elasticsearch:n aggregointiviitekehys (engl. aggregation framework) mahdollistaa datan analysoinnin erilaisten yhteenvetojen, ryhmittelyjen ja tilastojen avulla. Viitekehys perustuu aggregointeihin, joita voidaan yhdistellä kompleksisten summausten aikaansaamiseksi. Elasticsearch mahdollistaa kyselyiden suorittamisen ja kyselytulosten aggregoinnin samanaikaisesti. Aggregointeja on erityyppisiä ja ne voidaan jaotella bucketing, metric, matrix ja pipeline -aggregointeihin. [2] Tässä työssä hyödynnettiin bucketing ja

metric -aggregointeja. Bucketing-aggregoinnit ryhmittelevät dokumentteja määrättyjen kriteerien perusteella ja metric-aggregoinnit suorittavat matemaattisia operaatioita joukolle dokumentteja [2].

Elasticsearch mahdollistaa skriptien käytön erilaisiin operaatioihin. Sisäänrakennettuna oletuskielenä Elasticsearch:ssä on Painless. Painless-skriptejä voidaan käyttää kaikkialla, missä skriptien käyttö on mahdollista. Painless on yksinkertainen ja suojattu skriptikieli, joka on suunniteltu käytettäväksi Elasticsearch:in kanssa. Painless on Java-ohjelmointikielen osajoukko, johon on lisätty skriptikielien piirteitä helpottamaan skriptien kirjoittamista. Painless-skriptit käännetään suoraan Javan tavukoodiksi ja ne suoritetaan Java-virtuaalikoneessa. [8]

Kibana on selainpohjainen tiedon analysointi- ja visualisointialusta, jonka avulla Elasticseach:n indekseihin tallennettua dataa voidaan tarkastella. Kibana on toteutettu HTML:llä ja JavaScriptillä, ja se käyttää Elasticsearch:a REST-rajapinnan kautta. Kibana on suunniteltu toimimaan yhdessä Elasticsearch:n kanssa. Kibanan visualisoinnit perustuvat Elasticsearch:n kyselyihin ja aggregointeihin. [10]

Kibana-ohjelmistoa käytetään tässä työssä tiedon analysointiin ja tiedon esittämiseen päätöksentekijöille. Kibanaan voidaan tallentaa hakuja, visualisointeja ja muodostaa erilaisia kojetauluja (engl. dashboard) tarkkailtavista tiedoista.

6. TIEDONLOUHINNAN TOTEUTUS

Tässä kappaleessa kuvataan tarkemmin tiedonlouhinnan toteutus ja käytetyt menetelmät. Tilaajaorganisaation tavoitteena oli saada informaatiota asiointipalvelun nykytilasta. Kuvaileva tiedonlouhinta sopi tähän tarkoitukseen hyvin, sillä sen avulla voidaan kuvata miten asiat ovat tällä hetkellä tai miten ne ovat olleet aiemmin. Tässä työssä hyödynnettiin kuvailevan tiedonlouhinnan menetelmistä seuraavia: tietoaineiston tiivistäminen, visualisointi ja tietojen tarkastelua ajan suhteen.

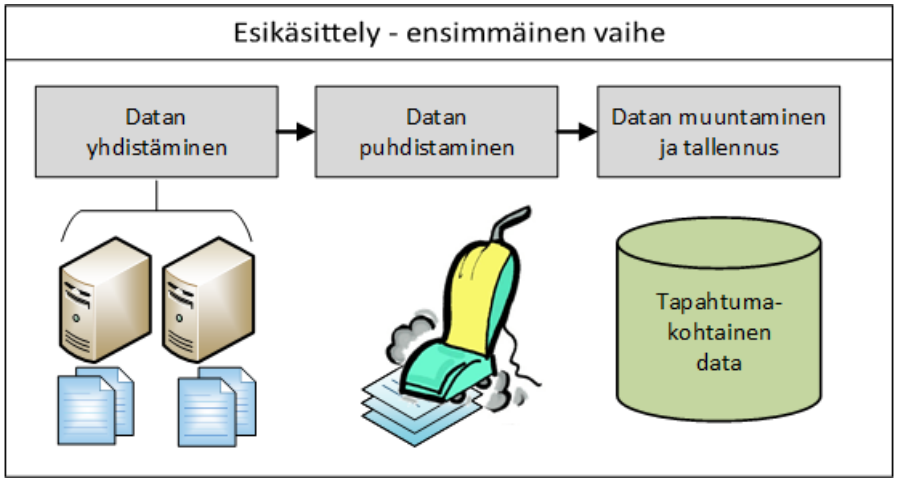
Tiedonlouhinnan tulokset esitellään kappaleessa 6.3. Toteutus noudatti pääpiirteissään kappaleessa 4.4 kuvattua käytön louhinnan -prosessia. Käytön louhinnan avulla weblokeilta saadaan palvelun käyttöön liittyvää informaatiota. Toteutettua järjestelmää testattiin toteutuksen eri vaiheissa, jotta järjestelmän luotettavuus ja tiedonlouhinnan tulosten oikeellisuus pystyttiin varmistamaan. Järjestelmän testaus on kuvattu kappaleessa 6.4.

Työssä pyrittiin huomioimaan kappaleessa 3.2 kuvatut weblokien hyödyntämiseen liittyvät haasteet. Verkkopalvelu on toteutettu yhden sivun arkkitehtuurilla, jossa WWW-palvelimella sijaitsevaa taustapalvelua käytetään tiedon hakemiseen ja tallentamiseen. Tästä johtuen WWW-palvelimella sijaitsevilla sovelluslokeilla näkyvät vain käyttöliittymän synnyttämät pyynnöt taustapalvelulle. Tiedon välittämiseen liittyviin taustapalvelupyyntöihin eivät HTTP-protokollan tilattomuus ja välimuistien käyttö verkon eri tasoilla pääse vaikuttamaan. Verkkopalvelu kirjaa sovelluslokeille taustapalvelukutsun mukana lähetetyn datan, jolloin lokeilta voidaan tarkastella dataa, joka tavallisesti välitettäisiin POST-pyyntönä ja jäisi kirjautumatta lokeille. Sähköisen asiointipalvelun käyttäminen vaatii kirjautumisen palveluun. Käyttäjän siirtyessä autentikointipalvelun kautta verkkopalveluun, saa verkkopalvelu tiedon, kenestä käyttäjästä on kyse. Näin verkkopalvelun lokeilta voidaan tunnistaa käyttäjä, vaikka käyttäjän tekemät pyynnöt tulisivat yhteisen välityspalvelimen kautta. Autentikointipalvelu huolehtii käyttäjien kirjautumisesta ja tarkistaa istunnon oikeellisuuden verkkopalvelun sitä pyytäessä. Tiedonlouhinnassa hyödynnettiin autentikointipalvelun osalta vain taustapalvelulle välittyviä kirjautumistietoja, joihin HTTP-protokollan tilattomuus, välimuistit ja yhteiset välityspalvelimet eivät vaikuta.

Työssä tietosuoja-asetukset ja henkilörekisterilaki on huomioitu lokitiedostojen esikäsittelyssä. Lokidatasta ei viedä Elasticsearch:iin sellaisia tietoja, joista käyttäjän voisi tunnistaa ilman ulkopuolista tietoa.

6.1 Weblokien esikäsittely

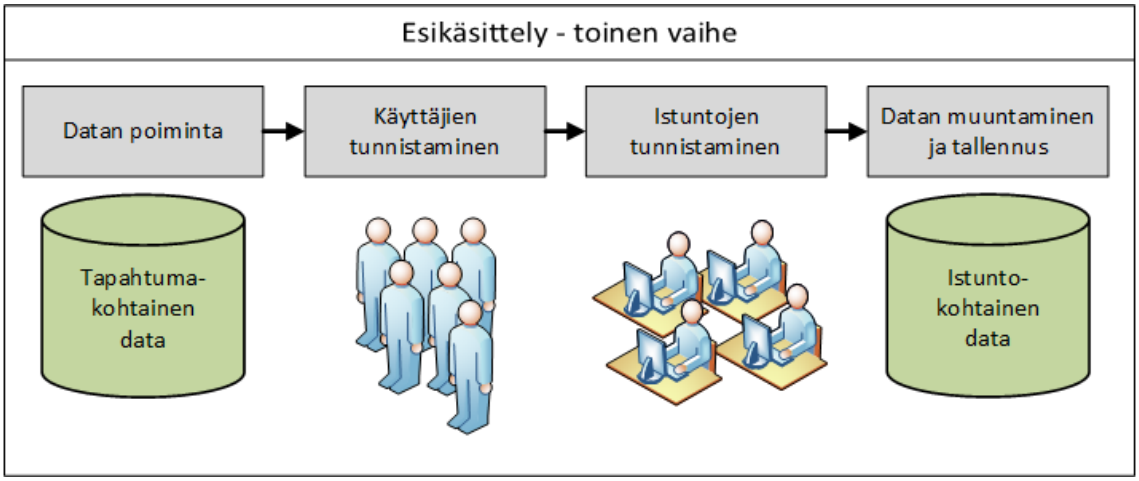
Weblokien esikäsittely jaettiin kahteen vaiheeseen. Ensimmäisessä vaiheessa lokidata eri sähköisen asiointipalvelun palvelimilta yhdistetään, dataan rikastetaan operatiivisen järjestelmän tiedolla, epäoleellisen pyynnöt poistetaan ja data anonymisoidaan. Lopuksi data muunnetaan haluttuun muotoon ja tallennetaan Elasticsearch:iin tapahtumakohtaisiin indekseihin (Kuva 10).



Kuva 10. *Esikäsittelyn ensimmäinen vaihe*

Ensimmäinen esikäsittelyn vaihe toteutettiin Filebeat, Logstash ja Elasticsearch -ohjelmistojen avulla. Muutoksia ja lisäyksiä tehtiin Filebeat:n ja Logstash:n konfiguraatiotiedostoihin. Filebeat-ohjelmiston konfiguraatiotiedosto oli nimeltään *Filebeat.yml*. Filebeat:a koskevat muutokset tehtiin kaikille sähköisen asiointipalvelun palvelimille. Logstash-ohjelmiston konfiguraatiotiedosto oli jaettu osiin, jokaiselle konfiguraatiotiedoston lohkolle oli oma tiedostonsa: sisäänmeno *00-input.conf*, suodin *10-filter.conf* ja ulostulo *20-output.conf*. Konfiguraatiotiedostoihin tehdyt muutokset kuvataan seuraavissa kappaleissa tarkemmin.

Esikäsittelyn toisessa vaiheessa tapahtumakohtaisista indekseistä poimitaan käyttäjien toimiin liittyvät lokitapahtumat. Tämän jälkeen lokitapahtumista tunnistetaan käyttäjät ja istunnot. Lopuksi tunnistetut istunnot tallennetaan istuntokohtaista dataa sisältäviin indekseihin (Kuva 11). Istuntojen tunnistus ja istuntokohtaisen datan muodostus toteutettiin Python-ohjelmointikielen ja Painless-skriptikielen avulla. Käyttäjien ja istuntojen tunnistus kuvataan tarkemmin kappaleessa 6.1.4.



Kuva 11. *Esikäsittelyn toinen vaihe*

6.1.1 Datan yhdistäminen

Tilaajaorganisaation sähköinen asiointipalvelu on hajautettu usealle palvelimelle ja tiedonlouhintaa varten eri palvelimien lokidata yhdistetään Filebeat ja Logstash-ohjelmistojen avulla. Filebeat-ohjelmistot lähettävät palvelimien lokitiedot Logstash-ohjelmistolle.

Autentikointipalvelun *application.log*-tiedoston lokidataa rikastetaan operatiivisen järjestelmän tiedoilla Logstash-ohjelmiston suodinvaiheessa. Lokitapahtuman koskiessa palveluun kirjautumista, haetaan sähköisen asiointipalvelun MongoDB-tietokannasta käyttäjätunnusta vastaava asiakaskohtainen tunnus. Lokidatan rikastaminen toteutettiin Logstash:n *jdbc_streaming*-lisäosan avulla (Kuva 12).

```
1   if [username]{
2      jdbc_streaming {
3        jdbc_driver_library => "/tmp/mongodb_jdbc.jar"
4        jdbc_driver_class => "com.dbschema.MongoJdbcDriver"
5        jdbc_connection_string => "jdbc:mongodb://db_user:db_password@host/db"
6        statement => 'db.collection.find(:stm).sort({_id:-1}).limit(1)'
7        parameters => { 'stm' => '{"uname":"%{username}"}'}
8        target => "user_data"
9      }
10  }
```

Kuva 12. *10-filter.conf: lokidatan rikastus MongoDB:stä*

Jdbc_streaming-lisäosa mahdollistaa SQL-kyselyiden suorittamisen tietokantoihin JDBC-ajurin avulla. Lisäosa suorittaa tietokantakyselyn ja tallentaa tuloksen *target*-osiossa määriteltyyn kenttään. *jdbc_driver_library, jdbc_driver_class* ja *jdbc_connection_string* -asetukset kertovat, missä käytetty JDBC-kirjasto sijaitsee, mitä jdbc-kirjaston luokkaa käytetään ja miten yhteys tietokantaan muodostetaan. Asetukset *statement* ja

parameters sisältävät kyselyn, joka suoritetaan ja sen parametrit. Jos kyselyn suoritus ei onnistu tapahtumaan lisätään *_jdbcstreamingfailure*-tunniste. [7]

6.1.2 Datan puhdistaminen

Datan puhdistuksessa lokitiedoista poistetaan epäolennaiset pyynnöt. Epäolennaisina pyyntöinä suodatettaan pois pyynnöt HTML-sivun lisäresursseihin. Filebeat-ohjelmistot eivät lähetä eteenpäin lokirivejä, joilta löytyy kuvien, tyylitiedostojen tai skriptitiedostojen tiedostopääte (Kuva 13: rivi 15).

```
1  filebeat.prospectors:
2  - input_type: log
3    paths:
4      - /polku/access.log
5    document_type: autentikointi-access
6    multiline.pattern: '^[0-9]{2}:[0-9]{2}:[0-9]{2}.[0-9]{3}'
7    multiline.negate: true
8    multiline.match: after
9
10
11 - input_type: log
12   paths:
13     - /polku/application.log
14   document_type: autentikointi-application
15   exclude_files: [".css", ".jpg", ".jpeg", ".png", ".js"]
16   multiline.pattern: '^[0-9]{2}:[0-9]{2}:[0-9]{2}.[0-9]{3}'
17   multiline.negate: true
18   multiline.match: after
```

Kuva 13. *Filebeat.yml: access.log ja application.log -tiedostojen prospector-osiot*

Lokidata anonymisoidaan ennen sen tallentamista Elasticsearch:iin. Jos lokitapahtuma sisältää lomaketietoja, käydään lomakedata läpi ja anonymisoitaviksi määritettyjen kenttien arvot hävitetään. Anonymisoinnin toteutuksessa hyödynnettiin *ryby*-lisäosaa (Kuva 14).

```
if [form_data]{
   ruby {
      code => "
        event.get('[form_data]').each { |key, value|
            if ( key =~ /(nimi|nimet|email|osoite|puhelinnumero)/i && value!='')
                event.set('[form_data]['+key+']', key)
            end
        }
        "
   }
}
```

Kuva 14. *10-filter.conf: lomakedatan anonymisointi*

6.1.3 Datan muuntaminen ja tallennus

Sähköisen asiointipalvelun lokitiedostot ovat sisältöformaatiltaan erilaisia. Erimuotoisten lokitapahtumien jäsentelyä varten lokitapahtumiin lisätään tunniste, joka ilmaisee minkä lokitiedoston tapahtumasta on kyse. Tunniste lisätään Filebeat-ohjelmistoissa ennen tapahtuman lähettämistä Logstash:iin (Kuva 13: rivit 5 ja 14). Tunnisteen lisääminen yksittäiselle lokitiedostolle vaati Filebeat-ohjelmien konfigurointitiedostoihin oman *prospectors*-osion jokaiselle lokitiedostolle.

Logstash vastaanottaa Filebeat-ohjelmien lähettämät lokitapahtumat ja jäsentelee ne eri suodinlisäosien avulla. Jäsentelyssä hyödynnetään tapahtumaan Filebeat:ssa lisättyä tunnistetta ja grok, mutate, kv, ruby, json ja date -lisäosia. Jokaisen lokitiedoston tapahtumia käsitellään omissa ehtolohkoissaan (Kuva 15).

```
filter {

    if [type] == "verkkopalvelu-access"{
       grok {
         ...
       }
    }

    if [type] == "verkkopalvelu-application"{
      grok {
         ...
       }
    }
}
```

Kuva 15. *10-filter.conf: lokitiedostokohtaiset ehtolohkot*

Kun lokirivin tiedot on ensin grok-lisäosan avulla jäsennelty kenttiin, muotoillaan kenttien tietoja mutate, kv, ruby, json ja date -lisäosien avulla. Tapahtuman kenttiä poistetaan, lisätään, nimetään uudelleen ja kenttien tietotyyppejä konvertoidaan mutate-lisäosan avulla. URL pyyntöjen parametrit jäsennellään ja lisätään tapahtumaan kv-lisäosassa. Ruby-lisäosan avulla poimitaan talteen merkkijonon osia sekä anonymisoidaan lokidataa edellisessä kappaleessa kuvatulla tavalla. JSON-muotoinen lomakedata jäsennellään ja

lisätään tapahtumaan json-lisäosaa hyödyntäen. Tapahtuman aikaleima luodaan yhdistelemällä kuluva päivämäärä lokirivin kellonaikaan date-lisäosassa.

Lokitiedostojen mahdollinen monirivisyys huomioidaan Filebeat ja Logstash -ohjelmistoissa. Ennen tapahtuman välittämistä eteenpäin Filebeat yhdistää moniriviset tapahtumat yhdeksi tapahtumaksi (Kuva 13: rivit 6-8 ja 16-18). Riveillä 6 ja 16 määritellään kaava, jolla lokirivin pitää alkaa, jos lokirivin alku ei vastaa kaavaa, lisätään lokirivi edeltävään riviin, jolta kaava löytyy. Filebeat lisää tapahtumaan *multiline*-tunnisteen tiedoksi tapahtuman monirivisyydestä. Logstash:ssa lokitapahtuman monirivisyys huomioidaan suodinvaiheessa. Moniriviset tapahtumat jäsennellään grok-lisäosassa määrittelemällä kaavan eteen ”(?m)” (Kuva 16).

```
1  filter{
2      if "multiline" in [tags]{
3          grok {
4            match => ["message", "(?m)%{TIME:timestamp}%{GREEDYDATA:multiline}$"]
5      }
6  }
```

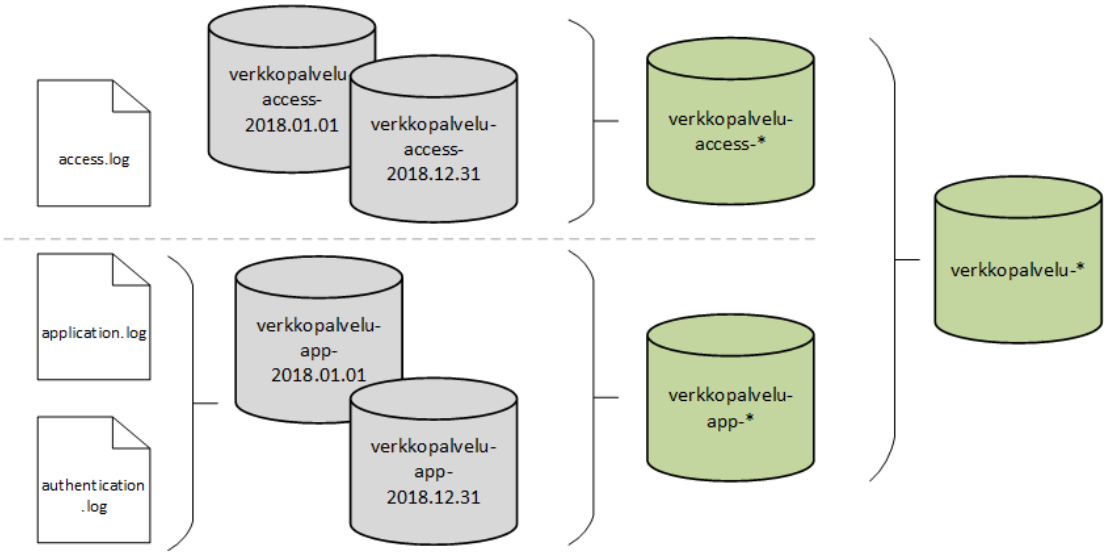
Kuva 16. *10-filter.conf: monirivisen tapahtuman käsittely*

Lokitapahtumien jäsentelyn jälkeen Logstash lähettää tapatumat Elasticsearch:lle. Logstash:n ulostulolohkoon määriteltiin jokaiselle lokitiedostolle oma ehtolohko (Kuva 17).

```
output {
    if [type] == "autentikointi-access"{
        elasticsearch{
            hosts => [ "elastic:9200" ]
            index => "autentikointi-%{+xxxx-ww}"
            document_type => "event"
        }

    }else if [type] == "autentikointi-application"{
        elasticsearch{
            hosts => [ "elastic:9200" ]
            index => "autentikointi-%{+xxxx-ww}"
            document_type => "event"
        }

    }else if [type] == "verkkopalvelu-access"{
        elasticsearch{
            hosts => [ "elastic:9200" ]
            index => "verkkopalvelu-access-%{+YYYY.MM.dd}"
            document_type => "event"
        }

    }else if [type] == "verkkopalvelu-application" {
        elasticsearch{
            hosts => [ "elastic:9200" ]
            index => "verkkopalvelu-app-%{+YYYY.MM.dd}"
            document_type => "event"
        }

    }else if[type] == "verkkopalvelu-authentication" {
        elasticsearch{
            hosts => [ "elastic:9200" ]
            index => "verkkopalvelu-app-%{+YYYY.MM.dd}"
            document_type => "event"
        }
    }
}
```

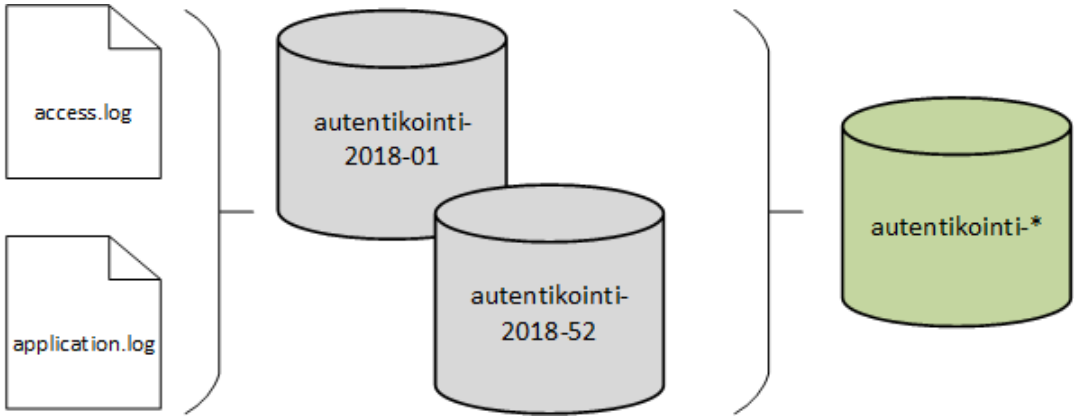
Kuva 17. *20-output.conf: ulostulot Elasticsearch:n indekseihin*

Tapahtumat tallennetaan Elasticsearch:ssa aikasidonnaisiin indekseihin, näin yksittäiset indeksit pysyvät järkevän kokoisina ja ajanhetkeen sidotut kyselyt kohdistuvat vain osaan indekseistä. Verkkopalvelun lokitapahtumat tallennetaan päiväkohtaisiin indekseihin. Koska verkkopalvelun *access.log* sisältää käyttäjien toimiin liittyviä tapahtumia, sen tapahtumat viedään omaan indeksiin jatkokäsittelyn helpottamiseksi. *Application.log* ja *authentication.log* -tiedostojen tapahtumat tallennetaan samaan indeksiin kts. Kuva 18.



Kuva 18. *Verkkopalvelun lokitapahtumien tallennus Elasticsearch:n indekseihin*

Autentikointipalvelun osalta lokidataa oli sen verran vähän, että lokitapahtumat päätettiin tallentaa viikkokohtaisiin indekseihin. Autentikointipalvelun lokitiedostojen lokitapahtumat viedään yhteiseen indeksiin kts Kuva 19.



Kuva 19. *Autentikointipalvelun lokitapahtumien tallennus Elasticsearch:n indeksiin*

Elasticsearch:ssa operaatioita pystytään kohdentamaan useisiin indekseihin korvausmerkkien avulla. Käyttämällä *verkkopalvelu-** ja *autentikointi-** -merkintöjä voidaan operaatioita kohdentaa indekseihin sovelluskohtaisesti (Kuva 18 ja 19). [2]

Elastisearch luo indeksin dokumenteille automaattisesti kuvauksen, kun dokumentteja tallennetaan indeksiin. Suoritettaessa kyselyitä useaan indeksiin ei indekseissä saa olla samannimisille kentille useita tietotyyppimäärityksiä. Jotta aikaperusteisten indeksien kentät saisivat yhdenmukaiset kuvaukset, käytettiin kuvausten luontiin indeksimalleja (engl. index template). Elasticsearch käyttää indeksimalleja automaattisesti, kun mallia vastaava uusi indeksi luodaan. [2]

```
PUT /_template/verkkopalvelu-app_template
{
    "template" : "verkkopalvelu-app*",
    "order" : 0,
    "version": 1,
    "mappings": {
      "event":{
        "properties":{
          "loglevel":{"type": "keyword", "index":true},
          "class":{"type": "keyword", "index":true}
          ...
        }
      }
   }
}
```

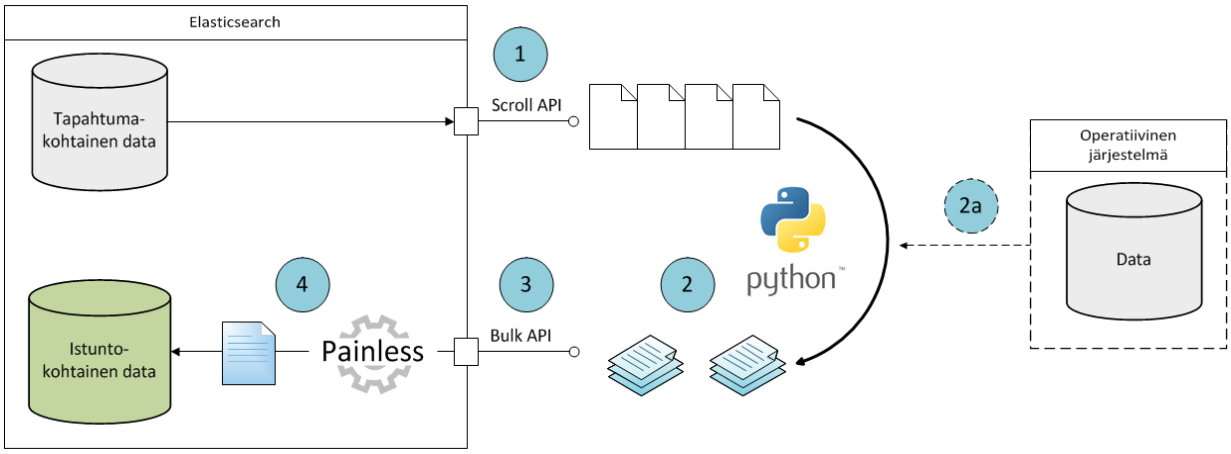
Kuva 20. *Elasticsearch: indeksimalli verkkopalvelu-app*-indekseille*

Indeksimalli luodaan lisäämällä uusi malli _template API:n kautta (Kuva 20). Kuvan indeksimallin nimi on *verkkopalvelu-app_template* ja mallia käytetään, kun luotavan indeksin nimi vastaa *template*-kentän arvoa. Työssä jokaiselle tapahtumakohtaiselle indeksille luotiin oma indeksimalli.

6.1.4 Käyttäjien ja istuntojen tunnistus

Käyttäjien ja istuntojen tunnistuksessa käytetään *verkkopalvelu-access*-indeksien tapahtumia, jonne sähköisen asiointipalvelun käyttäjien toimista syntyneet lokirivit tallentuvat. Verkkopalvelusovellus sisältää tiedot käyttäjän toimista käyttäjän kirjauduttua sähköiseen asiointipalveluun. Sovelluksen lokitiedot sisältävät asiakaskohtaisen tunnuksen ja istuntotunnisteen. Näiden tietojen avulla käyttäjät ja yksittäiset istunnot pystyttään helposti tunnistamaan, eikä erillistä istunnontunnistusmenetelmää tarvinnut käyttää.

Istuntokohtainen data kerätään tapahtumakohtaisesta datasta omiin indekseihin Python-ohjelmointikielen ja Painless-skriptikielen avulla. Samalla dataa rikastettaan operatiivisen järjestelmän tiedoilla (Kuva 21). Elasticsearch tarjoaa useita erilaisia ohjelmointirajapintoja. Istuntokohtaisen tiedon kokoamisessa hyödynnettiin Scroll ja Bulk -rajapintoja. Scroll API:n kautta Elasticsearch:sta voidaan hakea yhdellä kyselyllä suuria määriä dokumentteja [2]. Bulk API mahdollistaa useiden operaation toteuttamisen yhdellä rajapintakutsulla [2]. Python-ohjelmointikielelle on olemassa virallinen asiakaskirjasto Elasticsearch. Kirjasto sisältää apufunktiokokoelman helpers, jonka avulla Elasticsearch:n ohjelmointirajapintoja voidaan helposti hyödyntää. [21]



Kuva 21. *Istuntokohtaisen datan koonti:*
1. Tapahtumien haku scroll API:n kautta
2a. Datan rikastaminen
2. Tapahtumien ryhmittely ja Painless-skripti osion lisäys
3. Datan tallennus bulk API:n kautta
4. Istuntokohtaisten dokumentin muodostus

Tapahtumien haku scroll API:n kautta. Python-skripti hakee scroll-rajapinnan kautta Elasticsearch:n tapahtumakohtaisesta indeksistä kaikki kyselyä vastaavat tapahtumat, kyselyssä on mukana määritys tapahtuminen järjestyksestä. Scroll-rajapintaa voidaan käyttää helpers-kokoelmasta löytyvän *scan*-funktion avulla [21]. Scan on iteroituva ja palauttaa yksitellen kaikki scroll API:lle tehdyn pyynnön tulokset [21].

Datan rikastaminen. Tapahtumakohtaisia dokumentteja rikastetaan Python skriptissä hakemalla operationaalisesta tietokannasta ko. asiakasta koskevia tietoja. Tietojen hakemiseen käytetään Pythonin *JayDeBeApi*-moduulia, jonka avulla Python-skriptistä voidaan ottaa yhteys tietokantaa Javan JDBC-ajurin kautta [6].

Tapahtumien ryhmittely ja Painless-skripti osion lisäys. Scroll-rajapinnan kautta haetut tapahtumat käydään läpi ja dokumentit kootaan yhteen määriteltyjen tunnisteiden perusteella. Tapahtumista muodostetaan yksi toimenpidepyyntö, joka välitetään bulk API:lle. Pyynnössä tapahtumakohtaiset dokumentit ovat *params.event*-objektissa. Muodostettuun pyyntöön lisätään määritys Painless-skriptistä, jota käytetään vaiheessa 4 (Kuva 22: rivit 5-12).

```
1  {
2      "_op_type": "update",
3      "_id": document_id,
4      "scripted_upsert": True,
5      "script": {
6          "lang": "painless",
7          "file": "SessionUpdater",
8              "params": {
9                  "scriptMode": "incremental",
10                 "events": list(events)
11             }
12     },
13     "upsert": {}
14 }
```

Kuva 22. *Python-skriptin luoma toimenpidepyyntö bulk API:lle*

Datan tallennus bulk API:n kautta. Kun kaikki yhteenkuuluvat tapahtumat on käsitelty, lähetetään muodostettu toimenpidepyyntö Elasticsearch:n bulk-rajapintaan. Bulk-rajapintaa voidaan käyttää helpers-kokoelman *bulk*-funktion avulla [21]. Bulk-funktio saa parametrinaan generaattorin, joka palauttaa suoritettavat toimenpiteet (Kuva 23: rivi 3), bulk-funktio käy generaattoria läpi ja lähettää toimenpidepyynnöt Elasticsearch:lle.

```
1  elasticsearch.helpers.bulk(
2     es,
3     luo_toimenpide(),
4     chunk_size=1000,
5     request_timeout=30
6  )
```

Kuva 23. *Bulk-funktion käyttö*

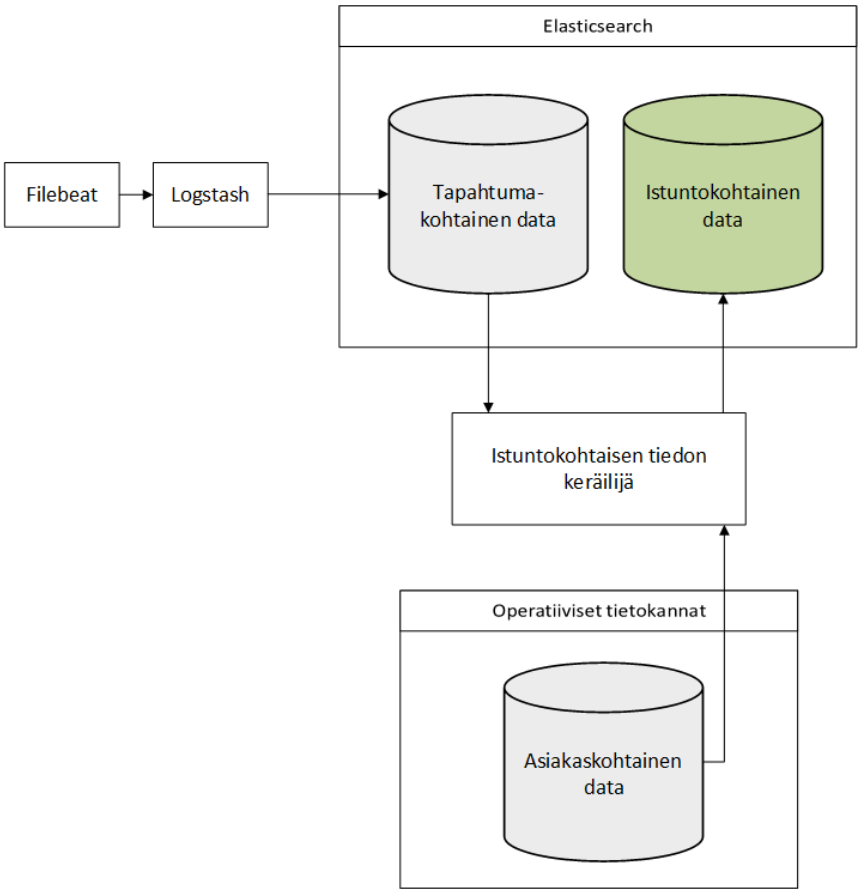
Bulk API:n kautta on mahdollista suorittaa index, create, delete ja update -operaatioita, toimenpidepyynnön *_op_type*-kenttä määrittelee suoritettavan operaation. Istuntokohtaisen datan tallennuksessa käytettiin update-operaatioita. Elasticsearch:n Update API mahdollistaa dokumenttien päivittämisen skriptien avulla. Update-operaatio hakee päivitettävän dokumentin indeksistä, suorittaa skriptin ja vie skriptin tuloksen takaisin indeksiin. Toimenpidepyynnön *script*-osiossa määritellään skriptitiedosto, käytetty skriptikieli ja skriptille välitettävät parametrit (Kuva 22: rivit 5-12). Indeksin dokumentteja voidaan päivittää update API:n kautta upsert-operaatioina. Tällöin jos päivitettävää dokumenttia ei löydy indeksistä, se lisätään sinne. Uusi dokumentti voidaan määritellä *upsert*-kentässä tai ajettavassa skriptissä. Kun uuden dokumentin sisältö määritellään skriptissä, asetetaan *scripted_upsert*-kentän arvoksi *true*. [2]

Istuntokohtaisen datan muodostuksessa bulk API:lle välitetyt update-operaatiot suoritetaan upsert-operaatioina niin, että määritelty Painless-skripti määrittää myös uuden dokumentin sisällön.

Istuntokohtaisen dokumentin muodostus. Painless-skripti huolehtii uuden dokumentin luomisesta tai vanhan dokumentin päivittämisestä. Painless-skripti käy *params.event*-objektissa listatut tapahtumat läpi ja muokkaa tapahtumat yhdeksi dokumentiksi, joka tallennetaan istuntokohtaiseen indeksiin. Painless-skripti poimii tapahtumadokumenteista vain tarpeelliset kentät ja laskee tapahtumien perusteella istuntokohtaiseen dataan liittyviä tunnuslukuja.

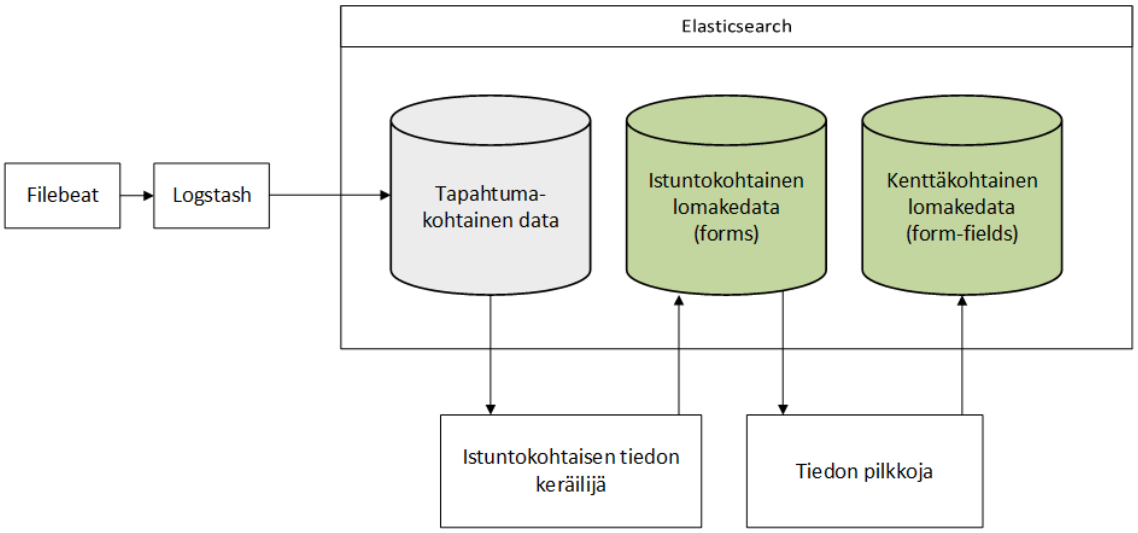
Työssä kootaan istuntokohtaista dataa käyttäjien istunnoista yleisesti ja istunnossa tapahtuneista lomakkeiden täytöstä tarkemmin. Istunto- tai lomakekohtainen data kootaan tapahtumapohjaisesta datasta omiin indekseihin edellä kuvatulla tavalla. Istunnot kootaan sessions-indeksiin ja lomakekohtainen data forms-indeksiin.

Sessions-indeksin dokumentit kootaan tapahtumakohtaisesta datasta asiakas- ja istuntotunnisteen avulla. Tapahtumakohtaiset dokumentit haetaan indeksistä asiakaskohtaisen tunnuksen, istuntotunnisteen ja tapahtuman aikaleiman mukaisessa järjestyksessä. Istuntokohtaista dataa rikastettaan operatiivisen järjestelmän tiedolla (Kuva 24). Operatiivisesta järjestelmästä haetaan asiakkaan sukupuoli, postinumero ja syntymävuosi. Näitä tietoja käytetään myöhemmin käyttäjien ryhmittelyyn ja karttapohjaisten visualisointien muodostamiseen. Painless-skripti laskee istunnolle keston ja käyttäjän iän syntymävuoden perusteella. Lisäksi painless-skripti määrittelee ne palveluosiot, joita istunnossa on käytetty. Palveluosioiden määrittely perustuu tiettyihin taustapalvelukutsuihin, joita tapahtuu vain tietyissä verkkopalvelun osioissa.



Kuva 24. *Istuntokohtaisen datan muodostus*

Forms-indeksin dokumentit kootaan tapahtumakohtaisesta indeksistä asiakaskohtaisen tunnuksen, istuntotunnisteen, lomakkeen asiointitunnuksen ja tapahtuman aikaleiman mukaisessa järjestyksessä. Painless-skripti laskee lomakkeelle sivu- ja kenttäkohtaisia täyttöaikoja, sekä määrättyjen kenttien lukumääriä. Lomakekohtainen data joudutaan kokoamisen jälkeen pilkkomaan osiin, jotta sisäkkäisten olioiden (engl. nested object) visualisointi Kibanassa olisi mahdollista (Kuva 25). Pilkkominen jouduttiin suorittamaan, koska käytössä ollut Kibana versio ei tue aggregointitoimenpiteiden suorittamista dokumenttien sisäisille olioille. Lomakekohtainen data pilkotaan osiin lomakkeen kenttien mukaisesti ja tallennetaan form-fields-nimiseen indeksiin. Sisäkkäisten olioiden pilkkominen omaan indeksiin toteutettiin Python-skriptillä hyödyntäen Elasticsearch:n scroll ja bulk-rajapintoja.



Kuva 25. *Lomakekohtaisen datan muodostus*

Koska tilaajaorganisaatio ei nähnyt reaaliaikaista käytön seurantaa tarpeellisena, päädyttiin istunto- ja lomakekohtainen data kokoamaan tapahtumakohtaisesta datasta omiin indekseihin kerran vuorokaudessa. Istunto- ja lomaketietojen muodostuksessa käytetyt Python-skriptit ajastettiin *crontab*-ohjelman avulla suoritukseen joka yö. Python-skriptit käsittelevät edellisen päivän *verkkopalvelu-access*-indeksiä kokonaisuudessa ja muodostavat siitä istuntokohtaisia dokumentteja sessions ja forms -indekseihin.

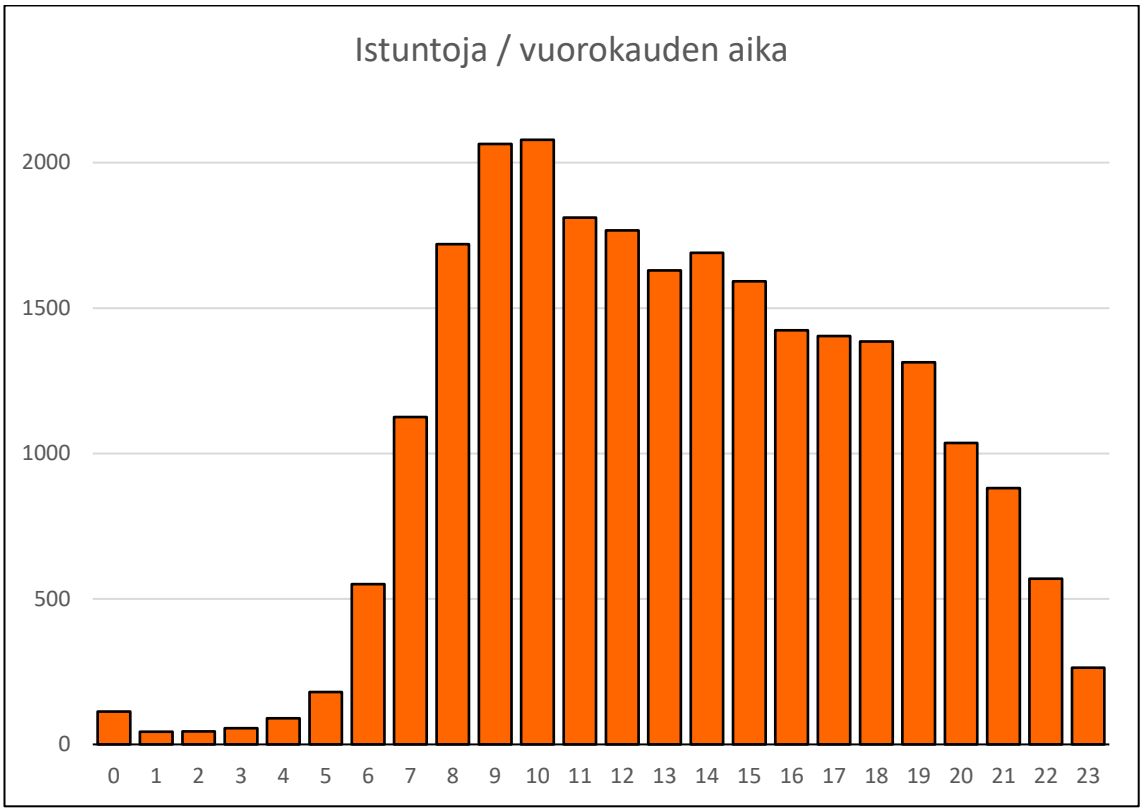
6.2 Kaavojen löytäminen

Tässä työssä kaavojenlöytämisvaiheessa käytettiin tilastollisia menetelmiä niiden yleisyyden ja tehtävään soveltuvuuden perusteella. Tietoaineiston summaaminen, visualisointi ja tarkastelu eri ajanhetkinä luo kuvan tietoaineiston yleisistä ominaisuuksia, joiden avulla voidaan muodostaa kuva asiointipalvelun käytöstä ja toiminnasta.

Kaavojen löytämisessä tietoaineistona käytettiin istuntokohtaistadataa indekseistä sessions ja forms sekä tapahtumakohtaistadata indekseistä verkkopalvelu-app-* ja autentikointi-*. Datasta muodostettiin frekvenssijakaumia, laskettiin tunnuslukuja (keski-, minimi- ja maksimiarvoja) ja muodostettiin erilaisia visualisointeja. Tunnuslukujen laskennassa hyödynnettiin Elasticsearch:n *avg, min, max* ja *sum* -aggregointeja.

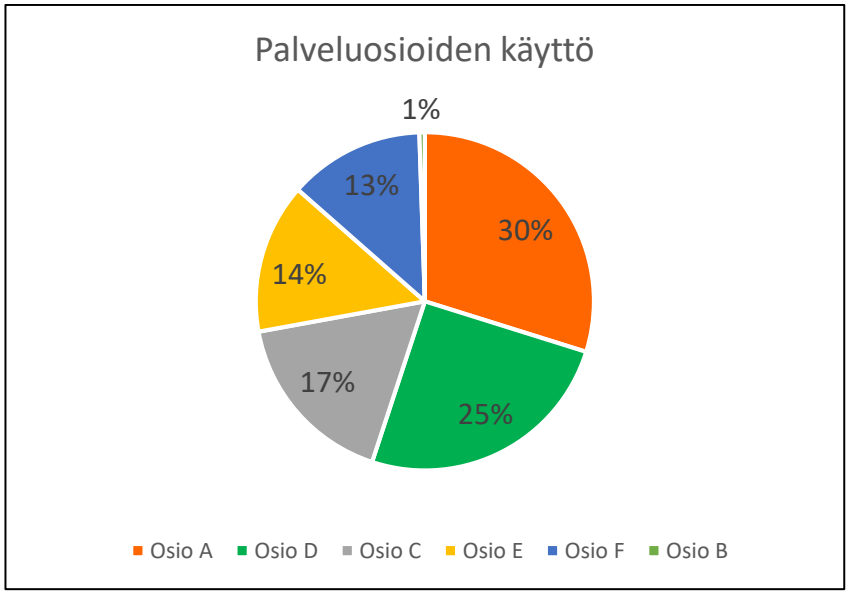
Elastiseach:n indekseissä sijaitsevaa dataan analysoitiin pääasiassa Kibanan avulla. Dataa tutkittiin ensin Kibanan discover-osiossa, minkä jälkeen datasta koottiin erilaisia visualisointeja. Kibanan Timelion-lisäosan avulla yhdistettiin eri indeksien dataa ja tehtiin vertailuja eri ajanhetkiin.

Istunnoista (indeksi: sessions) muodostettiin frekvenssijakaumia ajan, käyttäjän iän ja asuinalueen mukaan. Käyttäjiä jaoteltiin asiakastyypin mukaan yksityisiin ja yritysten edustajiin. Istunnon kestolle laskettiin keski-, minimi- ja maksimiarvoja.



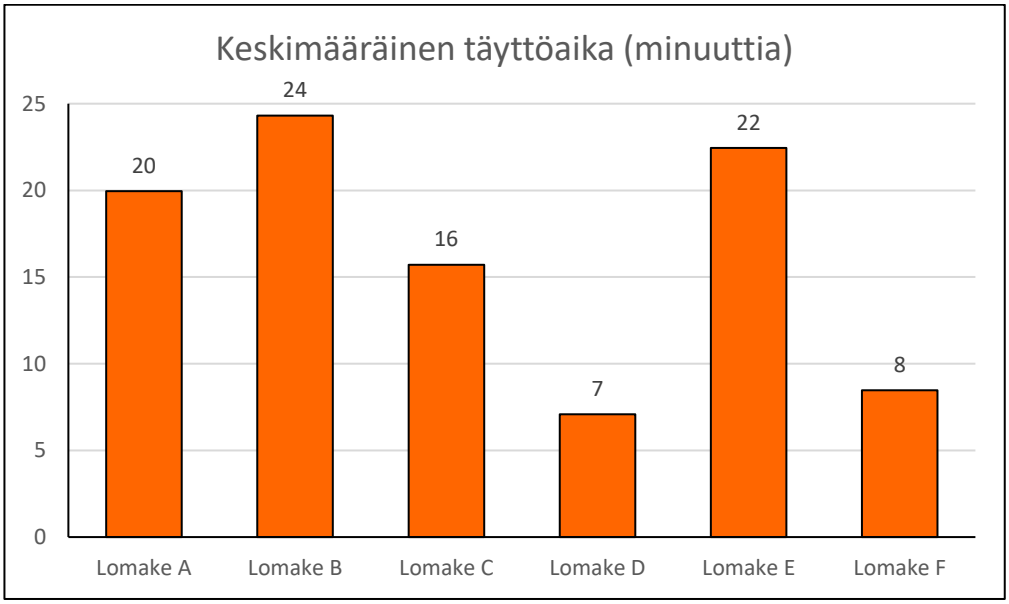
Kuva 26. *Istuntojen lukumäärä eri vuorokauden aikoina*

Istuntokohtaista dataa muodostettaessa istuntoihin tallennettiin palveluosiot, joissa käyttäjä istuntonsa aikana vieraili. Eri palveluosioiden käyttöä tarkasteltiin niiden prosenttiosuuksien avulla (Kuva 27).



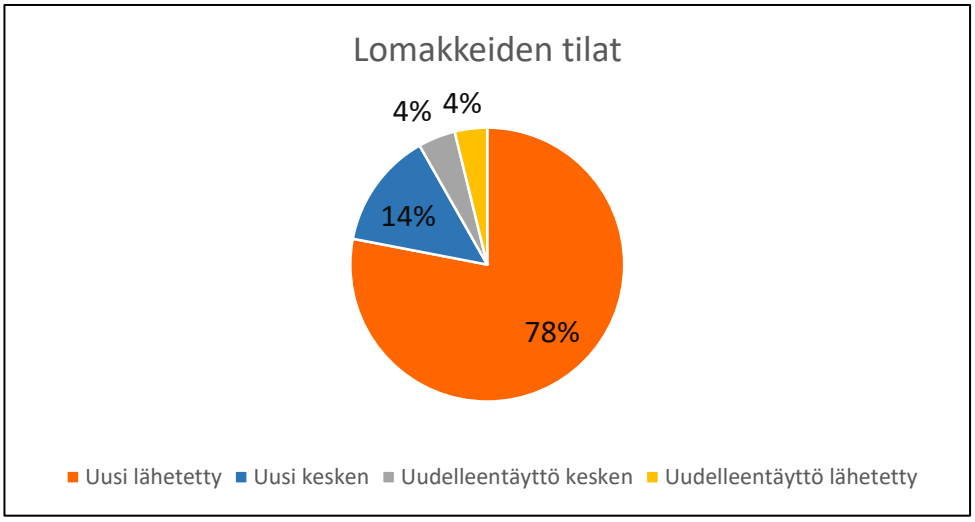
Kuva 27. *Palveluosioiden käytön jakautuminen*

Lomaketiedoista (indeksit: forms ja form-fields) laskettiin lomakekohtaisia ja lomakekenttäkohtaisia tunnuslukuja. Eri lomaketyypeistä muodostettiin frekvenssijakaumia ajan, lomaketyypin ja tilan mukaan. Lomaketietojen osalta keskityttiin muodostamaan kuva lomakkeen täytöstä ja siihen kuluvasta ajasta.



Kuva 28. *Keskiarvot eri lomaketyyppien täyttöön kuluvasta ajasta*

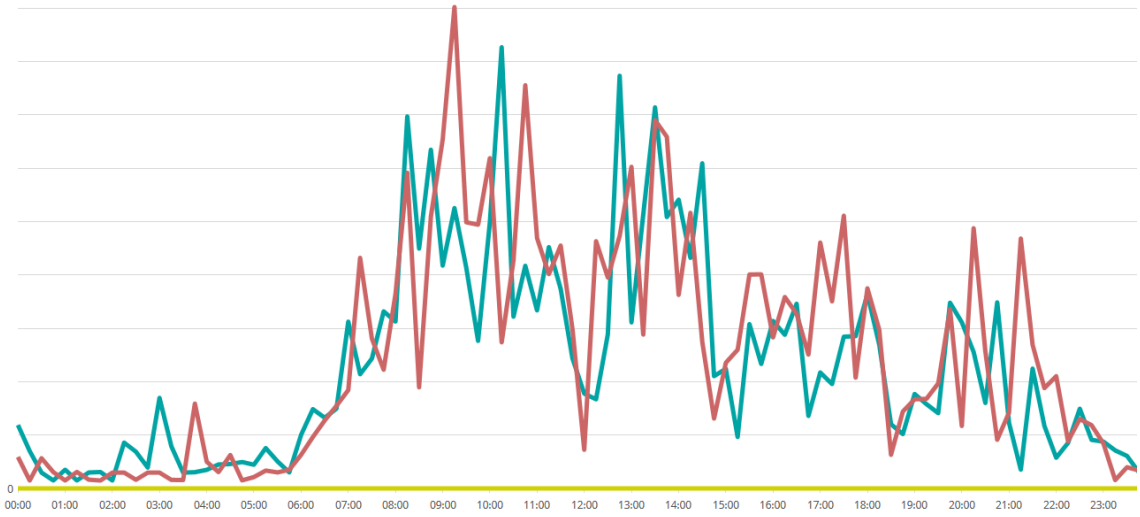
Lomakkeet tallentuvat automaattisesti sähköisessä asiointipalvelussa ja lomakkeen täyttöä voi jatkaa palvelussa myöhemmin. Lomakkeen täytölle määriteltiin neljä erilaista tilaa *uusi lähetetty*, *uusi kesken, uudelleentäyttö lähetetty* ja *uudelleentäyttö kesken*. Lomakkeiden eri tiloja tarkasteltiin niiden prosenttiosuuksien avulla (Kuva 29).



Kuva 29. *Lomakkeiden tilat*

Autentikointipalvelun tapahtumakohtaisestadatasta (indeksi: autentikointi-*) muodostettiin kirjautumiseen liittyviä visualisointeja ja tilastoja. Eri kirjautumistapoja kuvattiin frekvenssijakaumilla ja kirjautumisten lukumääriä tilastoitiin. Autentikointipalvelun toimintaa tarkasteltiin virheilmoitusten ja HTTP-tilakoodien avulla.

Verkkopalvelun toimintaan liittyvästä datasta (indeksi: verkkopalvelu-app-*) muodostettiin visualisointeja ja tilastoja, sekä laskettiin erilaisia tunnuslukuja. Taustapalvelukutsujen vastausajoille laskettiin keski-, minimi- ja maksimiarvoja. Eri palvelinten tapahtumien lukumääriä tarkkailtiin Timelion-visualisointien avulla, virheilmoituksista ja poikkeuksista muodostettiin tilastoja.



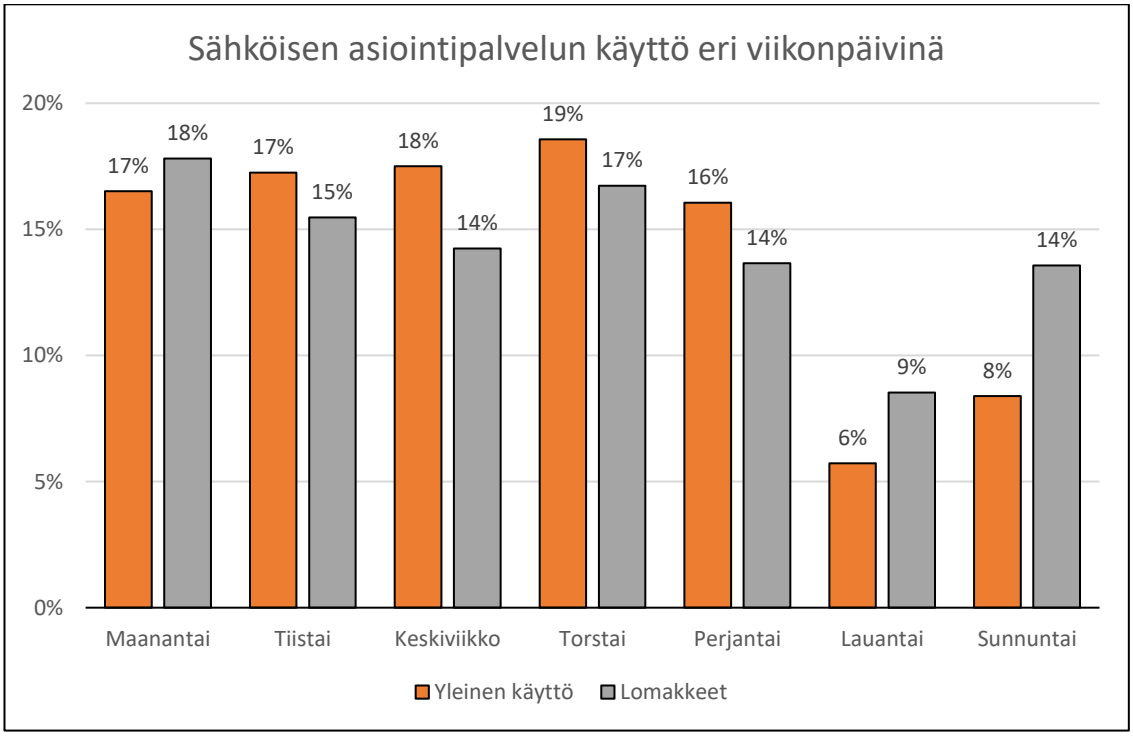
Kuva 30. *Palvelinten liikenne eri ajanhetkinä*

6.3 Kaavojen analysointi, työn tulokset

Kaavojen analysointi -vaiheessa löydetyistä kaavoista poimittiin kiinnostavat tilastot ja kaavat. Samalla kerättiin informaatiota sivuston käyttäjien käyttäytymisestä ja palvelun toiminnasta. Kaavojen analysointimenetelmänä käytettiin visualisointia. Visualisointi helpottaa kiinnostavien kaavojen tunnistamista ja sen avulla pystytään tuomaan esiin tietoaineiston trendejä. Kaavojen ja tilastojen kiinnostavuutta tarkasteltiin yhdessä eri organisaatioyksiköiden edustajien kanssa. Näin pystyttiin paremmin huomioimaan spesifisien päätöksentekijöiden tarpeita ja kiinnostuksen kohteita, sekä informaatiota pystyttiin tarkastelemaan kokonaisvaltaisesti eri organisaatioyksiköiden näkökulmista.

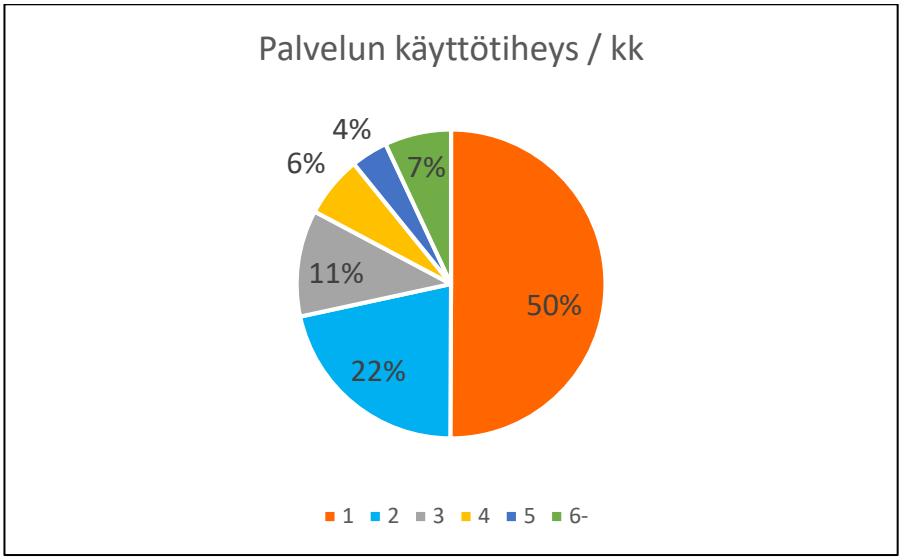
Kaavojen analysointi -vaiheessa pyrittiin huomioimaan kappaleessa 3.2 esiin nostetut weblokien luonteeseen liittyvät haasteet. Informaatiota kerättiin tarkastelemalla tietoainestoa eri ajanhetkinä ja tekemällä vertailuja. Yksittäisten tietojen tarkastelua vältettiin.

Sähköisen asiointipalvelun käytöstä huomattiin, että palvelun käyttö ajoittuu arkipäiviin, viikonloppuisin palvelua käytettään huomattavasti vähemmän (Kuva 31). Eniten käyttäjiä käy palvelussa klo 9 ja 10 välillä (Kuva 26). Palvelun käyttö jakaantuu eri osioiden välillä tasaisesti lukuun ottamatta osiota B, jota käytetään vain noin prosentissa istuntoja (Kuva 27).



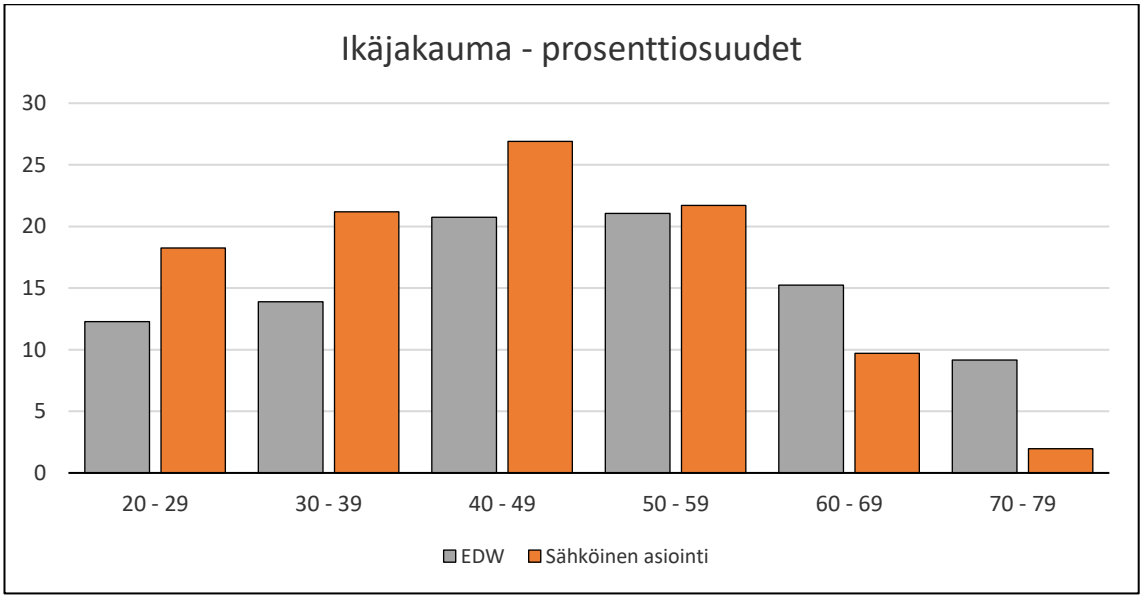
Kuva 31. *Sähköisen asiointipalvelun käyttö eri viikonpäivinä*

Sähköisen asiointipalvelun keskimääräinen istunto kestään noin 11 minuuttia ja valtaosa käyttäjistä käyttää palvelua 1-5 kertaa kuukaudessa (93%) (Kuva 32).



Kuva 32. *Palvelun käyttökertojen prosenttiosuudet*

Palvelun käyttäjiin liittyvää informaatiota verrattiin organisaation asiakkaisiin organisaation tietovarastosta saadun datan avulla. Verrattaessa asiakkaiden jakaumaan yritys- ja yksityisasiakkaisiin huomattiin sähköisen asiointipalvelun olevan suositumpi yksityisasiakkaisen keskuudessa. Käyttäjien ikäjakauma painottui nuorempiin asiakkaisiin, tämä vahvisti käsitystä sähköisen asioinnin kiinnostavuudesta nuorten keskuudessa.

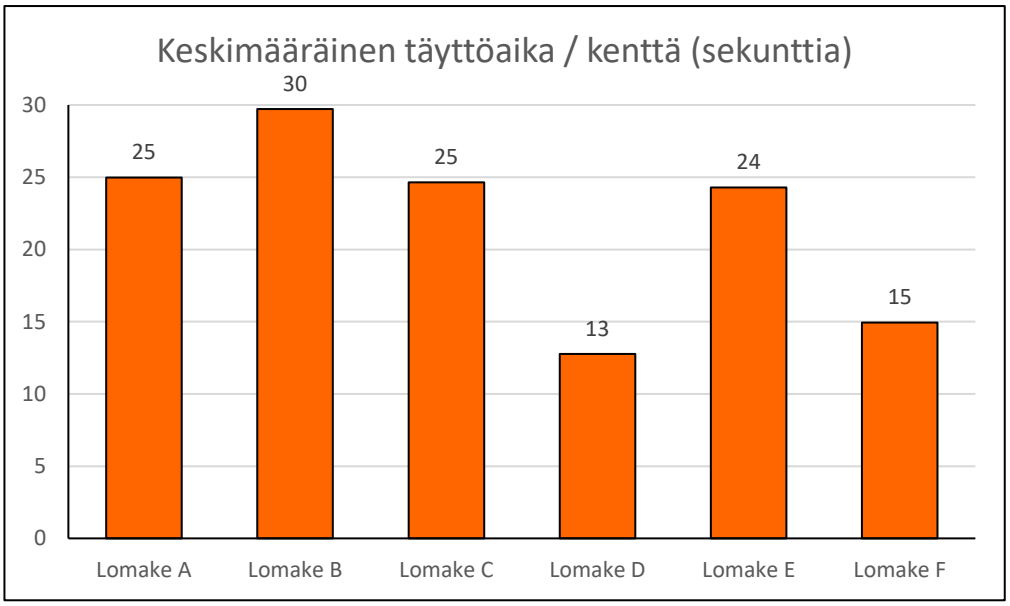


Kuva 33. *Käyttäjien ikäjakauma - sähköinen asiointi vs. tietovarasto*

Sähköisen asiointipalvelun käyttöön liittyvää informaatiota voidaan hyödyntää palvelun kehittämiseen liittyvässä päätöksenteossa ja markkinoinnissa. Käyttöajankohtiin liittyvää

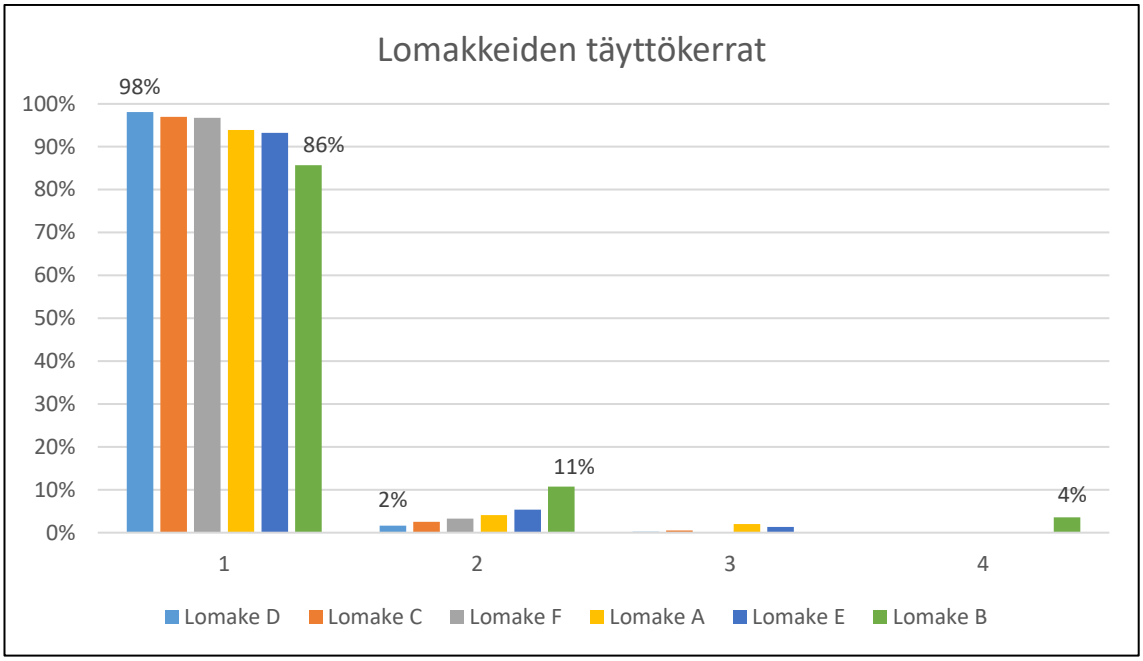
informaatiota voidaan hyödyntää resursoinnin suunnittelussa. Informaatiota voidaan hyödyntää esimerkiksi mietittäessä chat-ominaisuuden lisäämistä palveluun ja siihen liittyvän päivystyksen tarpeellisuutta eri ajankohtina. Palvelun eri osioiden käyttöasteinformaatiota voidaan hyödyntää, kun mietitään osioiden kehittämistarpeita ja osion tarpeellisuutta. Osion matala käyttöaste voi heijastaa osion vaikeaselkoisuutta tai hukkumista muiden osioiden joukkoon.

Lomakedatan osalta informaatiota kerättiin lomakkeiden täytöstä. Lomakkeiden täyttöajankohdat poikkesivat yleisestä asiointipalvelun käyttöajankohdista, lomakkeita täytetään tasaisemmin eri viikonpäivinä (Kuva 31). Lomakkeiden täyttöaika vaihteli suuresti riippuen lomakkeesta, tämä oli oletettavaa koska lomakkeet ovat eripituisia ja sisältävät erilaisia kysymyksiä. Lomakkeiden pituus huomioitiin laskemalla keskimääräinen täyttöaika lomakkeelle suhteutettuna lomakkeen kenttien lukumäärällä (Kuva 32).



Kuva 34. *Keskimääräinen lomakkeen täyttöaika suhteutettuna kenttien lukumäärään*

Lomakkeiden täyttöä tutkittiin sivu- ja kysymystasolla; näin löydettiin eniten aikaa vievät sivut ja kentät. Lomakedatasta tutkittiin myös lomakkeiden täyttökertoja. Suurin osa lomakkeista saadaan täytettyä yhdellä kerralla (86-98%), lomakkeesta riippuen täyttökertoja voidaan tarvita kuitenkin jopa neljä ennen kuin lomake saadaan lähetetyksi (Kuva 34). Lomakkeen keskimääräinen täyttöaika (Kuva 27) korreloi osittain käyttökertojen lukumäärään, lomake B:n keskimääräinen täyttöaika on pisin, samoin sen vaatimat täyttökerrat. Sama on havaittavissa lomakkeiden A ja E kohdalla.



Kuva 35. *Lomaketyyppien tarvitsemat täyttökerrat*

Sähköisen asiointipalvelun lomakkeiden täytöstä kerättyä informaatiota voidaan hyödyntää, kun tehdään päätöksiä lomakkeiden sisällöstä ja kehitetään lomakkeiden helppotäyttöisyyttä. Informaatiosta saadaan tukea siihen, mitä lomakkeita pitää kehittää ja muokata, kun halutaan vähentää lomakkeiden täyttöön kuluvaa aikaa ja täyttökertoja.

Palvelun toiminnasta löydettiin tilastojen avulla yleisimmät virheet ja poikkeukset. Tilastoja on tarkoitus käyttää virheiden korjaamiseen ja sovelluksen toiminnan parantamiseen. Verkkopalvelun taustapalvelukutsujen vasteajoista voitiin havaita, mitkä kutsut vievät eniten aikaa ja jatkossa näitä kutsujen pyritään optimoimaan. Sähköisen asiointipalvelun käytönseurannan avulla nähdään myös sopivimmat hetket palvelun suunnitelluille käyttökatkoille, jolloin käyttäjiä on mahdollisimman vähän palvelussa.

Tulevaisuudessa sähköiseen asiointipalveluun tehtyjä muutoksia voidaan seurata vertailemalla palvelun toimintaa eri ajanhetkinä. Näin voidaan havainnoida tehtyjen muutosten vaikutukset. Kun lomakkeisiin tehdään muutoksia, voidaan työssä toteutetun järjestelmän avulla seurata, miten muutokset vaikuttivat lomakkeiden täyttöön.

6.4 Järjestelmän testaus

Louhitun tiedon oikeellisuuden ja luotettavuuden varmistamiseksi tiedonlouhinta toteutusta testattiin projektin eri vaiheissa. Järjestelmä rakennettiin ensin testausympäristöön, jossa sen soveltuvuus projektin tavoitteisiin varmistettiin. Kun järjestelmä oli testausympäristössä tarpeellisella kattavuudella testattu, siirrettiin se tuotantoympäristöön.

Ennen tiedonlouhinnan aloittamista varmistettiin, että sähköinen asiointipalvelu kirjaa lokeihinsa työn tavoitteiden kannalta kaikki oleelliset tiedot. Varmistus suoritettiin tarkkailemalla sähköisen asiointipalvelun lokeja, kun palvelua käytettiin testausympäristössä.

Lokitiedostojen rivimääriä verrattiin Elasticsearch:n indeksien dokumenttimääriin, näin varmistettiin, että tarpeellisiksi määritellyt lokirivit saadaan talteen tiedonlouhintaa varten. Istunto- ja lomakekohtaisen datan muodostuksessa käytettyjä Python ja Painless -skriptejä testattiin vertailemalla yksittäisten istunto- ja lomakedokumenttien tietoja vastaaviin tapahtumakohtaisiin dokumentteihin. Tämän lisäksi lomakekohtaisia dokumentteja verrattiin operationaalisen järjestelmän tietoihin, jonne lomakkeet siirretään lähettämisen jälkeen. Näin varmistettiin, että lomakekohtainen data on Elasticsearch:ssa paikkansapitävää.

Toteutuksen aikana Logstash ja Elasticsearch -ohjelmistojen lokitiedostoja seurattiin, jotta mahdolliset virhetilanteet havaitaan ja saadaan korjattua mahdollisimman pian. Tapahtumien jäsentelyä Logstash:ssä tarkkailtiin etsimällä Elasticsearch:n tapahtumakotaisista indekseistä tapahtumia joiden *tags*-kenttään oli lisätty jokin virhetilannemerkinnöistä (_grokparsefailure, _jsonparsefailure, _jdbcstreamingfailure, _dateparsefailure).

7. JOHTOPÄÄTÖKSET

Onnistunut päätöksenteko organisaatioissa vaatii asianmukaisen ja luotettavan informaation hyödyntämistä päätöksenteon tukena. Organisaatioiden omat tietojärjestelmät sisältävät paljon informaatiota, jota voidaan oikeiden välineiden avulla hyödyntää päätöksenteossa.

Tämän työn tavoitteena oli vastata tutkimuskysymykseen: ”Miten weblokeja voidaan hyödyntää päätöksenteossa?”. Lähestymistavaksi valittiin tapaustutkimus, jossa hyödynnettiin olemassa olevaa kohdeorganisaation webpalvelua. Tilaajaorganisaatiolle toteutettiin järjestelmä, jonka kautta weblokeihin pohjautuva palvelun analysointi on mahdollista. Käyttäjien toimista ja palvelun toiminnasta louhittiin informaatiota, jota voidaan hyödyntää päätöksenteossa. Informaatio tarjoaa tukea päätöksiin, jotka liittyvät asiointipalvelun kehittämiseen tai siihen liittyviin resurssointeihin.

Weblokit ovat luonteelta haasteellinen tiedonlouhinnan lähde ja tämä on otettava huomioon informaatiota tarkasteltaessa. Tapahtumien vertailu eri ajanhetkillä on luotettavampaan kuin yksittäisten tapahtumien tarkastelu. Sähköisen asiointipalvelun lokitiedostoista louhitun tiedon luotettavuus paranisi huomattavasti, jos asiointipalvelun lokien kirjaamista tarkennettaisiin. Kaikkea tarpeellista tietoa ei ole saatavilla asiointipalvelun sovelluslokeista, vaikka palvelu pystyisi sen lokeihin kirjaamaan. Tämän vuoksi tiedonlouhinnassa ei kaikilta osin päästy parhaaseen mahdolliseen tulokseen tietojen luotettavuuden osalta. Ongelma ilmeni varsinkin palvelun eri osioiden käytön vertailussa. Työssä jouduttiin päättelemään käyttäjän istunnon aikana käyttämät palveluosiot määrätyistä taustapalvelukutsuista. Tästä johtuen käyttäjän viettämää aikaa eri palvelunosioissa ei saada selville eikä aikoja näin ollen voida vertailla.

Kun tiedonlouhintaa toteutetaan tarkoituksena tuottaa informaatiota päätöksenteon tueksi, on tärkeää, että projektissa on mukana tietotekniikan osaajien lisäksi myös eri instanssien päätöksentekijöitä. Näin projektin alusta asti on selvää mitä tietoa päätöksentekijät toivovat saavansa tiedonlouhinnan tuloksena. Toimialan ymmärtäminen on kriittistä onnistuneen tiedonlouhinnan ja tulosten analysoimisen kannalta. Tässä työssä vuoropuhelu organisaation eri yksiköiden edustajien kanssa toimi osittain hyvin. Edustajista osan kanssa yhteistyö jäi odotettua vähäisemmäksi resurssipulan ja henkilövaihdoksien vuoksi. Kaikkien päätöstentekijöiden tiiviimpi mukana olo olisi varmistanut, että louhittu tieto tarjoaa kaivattua tukea päätöksentekoon koko organisaatiossa.

Työssä toteutettua tiedonlouhintaa voitaisiin organisaatiossa jatkokehittää usealla tavalla. Tiedonlouhinnan tuloksista saataisiin tarkempaa ja luotettavampaa tietoa muutamilla lisäyksellä asiointipalvelun lokitukseen. Lisäämällä lokitukseen tieto siitä, mitä osiota

käyttäjä paraikaa käyttää, saataisiin tarkempaa tietoa siitä, mihin käyttäjien aika palvelussa kuluu. Työssä keskityttiin kahden sähköisen asiointipalvelun sovelluslokeihin. Työtä voisi laajentaa kattamaan sähköisen asiointipalvelun viestikeskussovellus sekä yhdistää tiedonlouhinnan tietoaineistoon myös WWW-palvelimen lokit asiointipalvelun osalta. Sähköisen asiointipalvelun lokeilta louhittu tieto olisi hyvä viedä organisaation tietovarastoon, jolloin sitä pystyttäisiin helpommin hyödyntämään ja yhdistelemään organisaation muuhun informaatioon.

Tapaustutkimuksen tulosten avulla voidaan todeta tiedonlouhinnan menetelmien soveltuvan weblokien hyödyntämiseen päätöksenteossa. Erityisesti tapaustutkimuksessa hyödynnetty käytön louhinta verkosta mahdollistaa kiinnostavan informaation etsimiseen verkon sekundäärisestä datasta mm. weblokeilta. Onnistut weblokien hyödyntäminen päätöksenteossa, vaatii weblokien luoteeseen liittyvien haasteiden huomioimisen ja tiedonlouhinnan toteuttamisen yhteistyössä päätöksentekijöiden kanssa. Weblokitiedon hyödyntäminen voi joissain tilanteissa vaatia muutoksia myös lokitietoa tuottavaan järjestelmään.

Tilaajaorganisaatiolla on käytössään useita muitakin järjestelmiä, joissa tässä työssä toteutettua tiedonlouhintaa voitaisiin hyödyntää. Toiveena on, että tiedonlouhinnan hyödyntäminen yhtenä päätöksenteon apuvälineenä organisaatiossa jatkuu ja kehittyy edelleen. Tiedonlouhinnan hyödyntämisen mahdollisuudet myös tulevien websovellusten yhteydessä kannattaa tiedostaa. Uusia sovelluksia suunniteltaessa olisi järkevää huomioida mahdollinen lokitietoon kohdistuva tiedonlouhinta ja suunnitella lokien kirjoitus alusta asti tiedonlouhinta huomioon ottaen. Lokien kirjaamisesta kannattaisi tehdä helposti parametrisoitava ja muunneltava, jotta se voidaan paremmin sovittaa myös tiedonlouhinnan tarpeisiin tulevaisuudessa.

LÄHTEET

[1] Beats Platform Reference, Elastic, verkkosivu. Saatavissa (viitattu 04.01.2018): https://www.elastic.co/guide/en/beats/libbeat/current/index.html.

[2] Elasticsearch Reference, Elastic, verkkosivu. Saatavissa (viitattu 02.01.2018): https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.

[3] EU:n tietosuojauudistus, Tietosuojavaltuutetun toimisto, verkkosivu. Saatavissa (viitattu 11.05.2018): http://tietosuoja.fi/fi/index/euntietosuojauudistus.html.

[4] Filebeat reference, Elastic, verkkosivu. Saatavissa (viitattu 02.01.2018): https://www.elastic.co/guide/en/beats/filebeat/current/index.html.

[5] Henkilötietolaki, 523, 1999. Saatavissa: https://www.finlex.fi/fi/laki/ajantasa/1999/19990523.

[6] JayDeBeApi 1.1.1, Python Software Foundation, verkkosivu. Saatavissa (viitattu 21.01.2018): https://pypi.org/project/JayDeBeApi/.

[7] Logstash Reference, Elastic, verkkosivu. Saatavissa (viitattu 02.01.2018): https://www.elastic.co/guide/en/logstash/current/index.html.

[8] Painless Scripting Language, Elastic, verkkosivu. Saatavissa (viitattu 20.01.2018): https://www.elastic.co/guide/en/elasticsearch/painless/current/index.html.

[9] R. Cooley, B. Mobasher, J. Srivastava, Web mining: information and pattern discovery on the World Wide Web, s. 558-567.

[10] B. Dixit, R. Kuc, M. Rogozinski, S. Chhajed, Elasticsearch: A Complete Guide, 1st ed. Packt Publishing, GB, 2017.

[11] M.A. Domingues, A.M. Jorge, C. Soares, S.O. Rezende, Web mining for the integration of data mining with business intelligence in web-based decision support systems, in: 2014, s. 120-136.

[12] S. Dumais, J. Teevan, R. Jeffries, D.M. Russell, D. Tang, Understanding user behaviour through log data and analysis, in: 2014, s. 349-370.

[13] M. Grönroos, Johdatus tilastotieteeseen: kuvailu, mallit ja päättely, in: Finn Lectura, Helsinki, 2003.

[14] J. Han, M. Kamber, Data mining: concepts and techniques, in: Morgan Kaufmann, San Francisco (CA), 2001, s. 5-28, 157-158, 179-217, 441-442.

[15] D. Hand, H. Mannila, P. Smyth, I. Books24x7, Principles of data mining, in: 1 ed., MIT Press, Cambridge (MA), 2001, s. 1-92.

[16] A. Hovi, H. Hervonen, H. Koistinen, Tietovarastot ja business intelligence, in: WSOYpro, Jyväskylä, 2009.

[17] M. Jafari, F. SoleymaniSabzchi, S. Jamali, Extracting Users´ Navigational Behavior from Web Log Data: A Survey, Journal of Computer Sciences and Applications, Vol. 1, No. 3, 2013, s. 39-45.

[18] M. Jennex, Big Data, the Internet of Things, and the Revised Knowledge Pyramid, ACM SIGMIS Database: the DATABASE for Advances in Information Systems, Vol. 48, No. 4, 2017, s. 69-79.

[19] Y. Jiang, Y. Li, C. Yang, E. Armstrong, T. Huang, D. Moroni, Reconstructing Sessions from Data Discovery and Access Logs to Build a Semantic Knowledge Base for Improving Data Discovery, ISPRS INTERNATIONAL JOURNAL OF GEO-INFORMATION, Vol. 5, No. 5, 2016.

[20] H. Kopackova, M. Škrobáčková, Decision support systems or business intelligence: what can help in decision making? 2006.

[21] H. Král, Python Elasticsearch Client, verkkosivu. Saatavissa (viitattu 21.01.2018): https://elasticsearch-py.readthedocs.io/en/master/index.html.

[22] P. Laininen, Tilastollisen analyysin perusteet, in: 3. korj. p. ed., Otatieto, Helsinki, 2004.

[23] M. Laukkanen, Päätöksenteko yrityksessä, in: Liiketaloustieteellinen tutkimuslaitos, Helsinki, 1968, s. 3-26.

[24] M. Mäntyneva, Tiedonlouhinta tukee asiakkuudenhallintaa. HAMK Unlimited: Professional, verkkosivu. Saatavissa (viitattu 20.03.2018): https://unlimited.hamk.fi/yrittajyys-jaliiketoiminta/tiedonlouhinta-tukee-asiakkuudenhallintaa/.

[25] M. Mäntyneva, Asiakkuudenhallinta, in: WSOY, Helsinki, 2001, s. 84-89.

[26] T. Markkula, A. Syväniemi, S. Suomela, Analytiikkamatka: datasta tietoon ja tiedolla johtamiseen, in: Suomen Liikekirjat, Helsinki, 2015, s. 63-73.

[27] D. Nagamalai, Web Log Data Analysis and Mining, in: Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, s. 459-469.

[28] D. Nicholas, P. Huntington, N. Lievesley, R. Withey, Cracking the code: web log analysis, Online and CD-Rom Review, Vol. 23, No. 5, 1999, s. 263-269.

[29] H. Paloheimo, M. Wiberg, Politiikan perusteet, in: Sanoma Pro Oy, Helsinki, 2012, s. 261-269.

[30] A.P. Sage, Decision theory, AccessScience, 2014.

[31] A. Salminen, Hallintotiede: organisaatioiden hallinnolliset perusteet, in: 9 ed., Edita Prima Oy, Helsinki, 2009, s. 79-88.

[32] V.L. Sauter, Decision Support Systems for Business Intelligence, Second Edition, in: Second ed., John Wiley & Sons, 2010, s. 3-66.

[33] H.A. Simon, The new science of management decision, Rev. ed. Prentice-Hall, Englewood Cliffs, 1977.

[34] H.A. Simon, Päätöksenteko ja hallinto, 1. p., 2. p. ed. Weilin & Göös, Espoo, 1979.

[35] J. Srivastava, R. Cooley, M. Deshpande, P. Tan, Web usage mining: discovery and applications of usage patterns from Web data, ACM SIGKDD Explorations Newsletter, Vol. 1, No. 2, 2000, s. 12-23.

[36] P. Sydänmaanlakka, Älykäs organisaatio: tiedon, osaamisen ja suorituksen johtaminen, in: 3. p. ed., Gummerrus Kirjapaino Oy, Jyväskylä, 2001, s. 171-196.

[37] Valtiovarainministeriö, Lokiohje, 2009.

[38] V. Verma, A.K. Verma, S.S. Bhatia, Comprehensive Analysis of Web Log Files for Mining, International Journal of Computer Science Issues (IJCSI), Vol. 8, No. 6, 2011, s. 199-199.

[39] Viestintävirasto, Lokien keräys ja käyttö, 2016.

[40] J.X. Yu, Y. Ou, C. Zhang, S. Zhang, Identifying interesting visitors through Web log classification, IEEE Intelligent Systems, Vol. 20, No. 3, 2005, s. 55-59.