Eemeli Mark

Käyttäjien toiminnan seuranta mikropalveluarkkitehtuurissa

Tieto- ja ohjelmistotekniikan kandidaatintutkielma

19. joulukuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Eemeli Mark

Yhteystiedot: eemeli.j.mark@student.jyu.fi

Ohjaaja: Annemari Auvinen

Työn nimi: Käyttäjien toiminnan seuranta mikropalveluarkkitehtuurissa

Title in English: User activity monitoring in microservice architecture

Työ: Kandidaatintutkielma

Opintosuunta: Tieto- ja ohjelmistotekniikka

Sivumäärä: 26+0

Tiivistelmä: Mikropalveluarkkitehtuuri aiheuttaa erityisiä haasteita käyttäjien toiminnan seurantaan erityisesti datan hajautumisen, käyttäjäkontekstin säilyttämisen ja lokienhallinnan osalta. Tutkimuksessa tarkastellaan näitä haasteita ja kartoitetaan tehokkaita ratkaisuja niiden voittamiseksi. Ratkaisut sisältävät hajautetun jäljityksen, käyttäjäkontekstin välittämisen ja keskitetyn lokienhallinnan. Työ tarjoaa suosituksia parhaista käytännöistä, jotka auttavat kehittäjiä toteuttamaan kokonaisvaltaisen käyttäjien toiminnan seurannan mikropalveluarkkitehtuurissa.

Avainsanat: mikropalveluarkkitehtuuri, käyttäjien toiminnan seuranta, hajautettu jäljitys, käyttäjäkontekstin välitys, lokienhallinta

Abstract: The microservice architecture presents challenges for user activity monitoring, particularly to data distribution, maintaining user context, and managing logs. The thesis examines these specific challenges and explores effective solutions to overcome them. The solutions include distributed tracing, user context propagation, and centralized log management. The thesis offers recommendations on best practices to help developers implement comprehensive user activity monitoring in microservice architecture.

Keywords: microservice architecture, user activity monitoring, distributed tracing, user context propagation, log management

Kuviot

Sisällys

1 Johdanto

Mikropalveluarkkitehtuuri on yleistynyt modernien ohjelmistojen kehittämisessä sen tarjoaman joustavuuden ja skaalautuvuuden ansiosta. Mikropalveluarkkitehtuuri on ohjelmistokehityksen lähestymistapa, jossa ohjelmisto koostuu joukosta pieniä, itsenäisiä palveluita, jotka kommunikoivat keskenään tarkasti määriteltyjen rajapintojen kautta (Newman 2021). Toisin kuin perinteiset monoliittiset järjestelmät, joissa koko ohjelmisto rakennetaan yhtenä kokonaisuutena, mikropalveluarkkitehtuuri jakaa ohjelmiston pienempiin erillisiin osiin.

Mikropalveluarkkitehtuurin hajautettu luonne tuo kuitenkin mukanaan haasteita erityisesti käyttäjien toiminnan seurantaan. Newman (2021) korostaa, että perinteisissä monoliittisissa järjestelmissä käyttäjien toiminnot voidaan helposti jäljittää ja analysoida, koska kaikki tapahtumat tapahtuvat yhdessä yhtenäisessä ympäristössä. Toisaalta Nadareishvili ym. (2016) huomauttavat, miten mikropalveluissa käyttäjän pyyntö saattaa kulkea useiden palveluiden läpi, mikä tekee kokonaiskuvan muodostamisesta haastavaa. Tätä näkemystä tukevat myös Villamizar ym. (2015), jotka painottavat hajautetun arkkitehtuurin monimutkaisuutta seurannan kannalta.

Käyttäjien toiminnan seuranta on olennaista palvelun laadun ja suorituskyvyn varmistamiseksi. Zhangin ja Shafiqin (2022) mukaan se mahdollistaa palvelun käytön analysoinnin, suorituskykyongelmien tunnistamisen, virheiden jäljittämisen sekä käyttäjäkokemuksen parantamisen. Lisäksi käyttäjien toiminnan seuranta on olennainen osa tietoturvaa, sillä poikkeavan käyttäytymisen havaitseminen auttaa ehkäisemään väärinkäytöksiä ja tietomurtoja.

Tässä tutkimuksessa pyritään syventämään ymmärrystä siitä, miten mikropalveluarkkitehtuuri vaikuttaa käyttäjien toiminnan seurantaan. Erityisesti tarkastellaan datan hajautumista, käyttäjäkontekstin säilyttämistä palveluiden välillä ja lokienhallintaa. Tavoitteena on kartoittaa tehokkaita ratkaisuja näiden haasteiden voittamiseksi keskittyen hajautettuun jäljitykseen, käyttäjäkontekstin välittämiseen ja keskitettyyn lokienhallintaan. Tutkimuksen tulokset auttavat kehittäjiä toteuttamaan kokonaisvaltaisen käyttäjien toiminnan seurannan mikropalveluarkkitehtuurissa, parantaen näin palveluidensa laatua, suorituskykyä ja turvallisuutta.

Luvussa 2 käsitellään mikropalveluarkkitehtuuri sekä siihen liittyvät hyödyt ja haasteet. Lu-

vussa 3 esitellään erityisiä haasteita, joita mikropalveluarkkitehtuuri aiheuttaa käyttäjien toiminnan seurannalle, kuten datan hajautuminen, käyttäjäkontekstin säilyttäminen ja lokienhallinta. Luvussa 4 tarkastellaan näiden haasteiden ratkaisuja, eli hajautettua jäljitystä, käyttäjäkontekstin välittämistä ja keskitettyä lokienhallintaa. Lisäksi luvussa esitellään kokonaisvaltainen käyttäjien toiminnan seurantamalli, joka yhdistää nämä ratkaisut yhtenäiseksi kokonaisuudeksi. Lopuksi luvussa 5 esitetään yhteenveto sekä pohditaan työn tulosten merkitystä ja mahdollisia jatkotutkimuksen suuntia.

2 Mikropalveluarkkitehtuuri

Mikropalveluarkkitehtuuri (engl. *microservice architecture*) pyrkii ratkaisemaan perinteisten monoliittisten järjestelmien ongelmia, erityisesti haasteita liittyen skaalautuvuuteen, ylläpidettävyyteen ja kehityksen nopeuteen. Newman (2021) määrittelee mikropalveluarkkitehtuurin ohjelmistokehityksen tyylinä, jossa ohjelmistot rakennetaan joukkona pieniä, itsenäisiä palveluita. Palvelut kommunikoivat keskenään tarkasti määriteltyjen rajapintojen kautta muodostaen yhdessä kokonaisen järjestelmän. Thönes (2015) täydentää määritelmää korostamalla, että mikropalvelut voidaan kehittää, testata, ottaa käyttöön ja skaalata itsenäisesti. Mikropalveluarkkitehtuuri onkin kasvattanut suosiotaan viime vuosina erityisesti suurten ja monimutkaisten järjestelmien kehittämisessä, joissa joustavuus, skaalautuvuus ja nopea kehityssykli ovat kriittisiä vaatimuksia. Tässä luvussa käsitellään mikropalveluarkkitehtuurin keskeisiä ominaisuuksia, sen tarjoamia hyötyjä sekä siihen liittyviä haasteita.

2.1 Mikropalveluarkkitehtuurin keskeiset ominaisuudet

Mikropalveluarkkitehtuurin toiminnallisuus perustuu sen erityisiin ominaisuuksiin, jotka mahdollistavat sen käytön laajoissa ja monimutkaisissa järjestelmissä. Näiden ominaisuuksien tarkastelu on tärkeää, jotta voidaan ymmärtää, miksi mikropalveluarkkitehtuurista on tullut suosittu valinta ohjelmistokehityksessä ja miten sen käyttö vaikuttaa järjestelmien suunnitteluun ja toteutukseen.

Newmanin (2021) mukaan yksi mikropalveluarkkitehtuurin tärkeä ominaisuus on palveluiden itsenäisyys. Hänen mukaansa jokainen palvelu toimii itsenäisenä kokonaisuutena, erillään muista järjestelmän osista. Toinen keskeinen piirre on yksittäinen vastuualue, kuten Dragoni ym. (2017) toteavat. Tämä tarkoittaa, että kukin mikropalvelu on vastuussa selkeästi rajatusta toiminnallisuudesta, mikä mahdollistaa palveluiden tarkan suunnittelun juuri niiden määriteltyihin tehtäviin.

Newman (2021) korostaa, että tarkasti määritellyt rajapinnat ovat keskeinen osa mikropalveluarkkitehtuuria. Niiden ansiosta palvelut voivat kommunikoida tehokkaasti, mikä tehostaa tiedonsiirtoa järjestelmässä. Alpers ym. (2015) painottavat teknologisen monimuotoisuuden

olevan keskeinen piirre mikropalveluarkkitehtuurissa, jolloin palvelut voivat olla toteutettu eri ohjelmointikielillä, alustoilla ja teknologioilla. Tämä lähestymistapa mahdollistaa sen, että jokaiselle tehtävälle voidaan valita parhaiten soveltuva työkalu.

Näiden ominaisuuksien ansiosta mikropalveluarkkitehtuuri tarjoaa lupaavan vaihtoehdon erityisesti järjestelmiin, jotka vaativat joustavuutta ja skaalautuvuutta. Se on suosittu ratkaisu monimutkaisissa ympäristöissä, joissa kyky mukautua muuttuviin tarpeisiin ja hyödyntää teknologian kehitystä on olennaista.

2.2 Hyödyt ja haasteet mikropalveluarkkitehtuurissa

Mikropalveluarkkitehtuurin käyttö tarjoaa merkittäviä hyötyjä verrattuna perinteiseen monoliittiseen arkkitehtuuriin. Huolimatta merkittävistä hyödyistä mikropalveluarkkitehtuuri tuo mukanaan myös haasteita.

Thönesin (2015) mukaan palveluiden itsenäisyys on yksi mikropalveluarkkitehtuurin keskeinen hyöty, koska muutokset yhdessä palvelussa eivät vaikuta suoraan järjestelmän muihin osiin. Tämä helpottaa palveluiden ylläpitoa ja vianmääritystä. Toisaalta Salah ym. (2016) huomauttavat, että tämä itsenäisyys lisää tietoturvan ja valvonnan haasteita, koska jokainen palvelu on suojattava erikseen, ja hajautetut ympäristöt voivat tehdä järjestelmän monitoroinnista monimutkaisempaa.

Dragonin ym. (2017) mukaan yksittäinen vastuualue auttaa palveluiden selkeyttämisessä ja helpottaa ylläpitoa ja kehitystä, sillä kukin palvelu vastaa tarkasti määritellyistä tehtävistä. Newmanin (2021) mukaan palveluiden välinen kommunikointi rajapintojen kautta vähentää järjestelmän riippuvuuksia. Tämä on kuitenkin mikropalveluille ominainen haaste, sillä rajapintojen suunnittelu ja ylläpito vaativat erityistä huomiota toisin kuin monoliittisessa järjestelmissä, joissa komponentit toimivat yhtenä kokonaisuutena ilman erillisiä rajapintoja.

Teknologinen monimuotoisuus tarjoaa tiimeille vapauden innovoida ja hyödyntää uusimpia teknologioita ilman, että koko järjestelmän täytyy siirtyä uusiin ratkaisuihin. Alpers ym. (2015) korostavat, että tämä mahdollistaa erilaisten teknologioiden rinnakkaisen käytön, mikä voi olla hyödyllistä erilaisten suorituskykytarpeiden täyttämisessä. Toisaalta Salah

ym. (2016) mainitsevat, että tämä tuo samalla monimutkaisuutta tiimeihin, sillä tällöin tiimien täytyy hallita enemmän eri teknologioita ja yhdistää ne toimivaksi järjestelmäksi. Tämä moniteknologinen ympäristö voi aiheuttaa haasteita kehityksessä, testauksessa ja ylläpidossa, koska eri teknologioiden yhteensovittaminen vaatii lisäresursseja ja osaamista.

Boncea, Zamfiroiu ja Bacivarov (2018) korostavat, että mikropalveluarkkitehtuurin joustavuus tekee siitä erityisen houkuttelevan vaihtoehdon suurille ja monimutkaisille järjestelmille. Joustavuus ilmenee heidän mukaansa siinä, että mikropalvelut mahdollistavat nopeamman kehityssyklin, koska tiimit voivat työskennellä itsenäisesti omien palveluidensa parissa ilman, että muutokset vaikuttavat koko järjestelmään. Monoliittisissa järjestelmissä yhden osan muutokset voivat vaikuttaa koko järjestelmään toimintaan, mikä Thönesin (2015) mukaan tekee kehityksestä ja sen käyttöönotosta monimutkaisempaa.

Villamizar ym. (2015) korostavat, että mikropalveluarkkitehtuurin etuna on niiden skaalautuvuus, joka saavutetaan, kun yksittäisiä palveluita voidaan skaalata itsenäisesti tarpeen mukaan. He mainitsevat, miten monoliittisten ohjelmistojen skaalautuminen on usein haastavaa, sillä ne saattavat sisältää monia palveluita, joista osa on käytetympiä kuin toiset. Tämä johtaa tilanteeseen, jossa koko järjestelmä on skaalattava vastaamaan yksittäisten komponenttien kasvavaa kysyntää, mikä ei ole kustannustehokasta.

Nadareishvili ym. (2016) mainitsevat kestävyyden ja vikasietoisuuden olevan keskeisiä hyötyjä mikropalveluarkkitehtuurissa. Yksittäisen palvelun vikaantuminen ei välttämättä vaikuta koko järjestelmän toimintaan, mikä parantaa järjestelmän kokonaiskestävyyttä toisin kuin monoliittisessa järjestelmässä. Tämä kuitenkin tuo mukanaan haasteita vikojen jäljittämisessä ja monitoroinnissa, sillä viat voivat olla hajautuneet useisiin eri palveluihin, mikä vaikeuttaa niiden paikantamista.

Näin ollen mikropalveluarkkitehtuuri tarjoaa mahdollisuuden kehittää joustavia ja skaalautuvia järjestelmiä, jotka voivat vastata nopeasti liiketoiminnan muuttuviin tarpeisiin. Sen käyttöönotto vaatii kuitenkin kykyä hallita monimutkaisuutta, joka liittyy hajautettuihin järjestelmiin.

3 Käyttäjien toiminnan seurannan haasteet

Tässä luvussa käsitellään mikropalveluarkkitehtuurin erityisiä haasteita käyttäjien toiminnan seurantaan liittyen. Ensin käsitellään käyttäjien toiminnan seuranta, jotta ymmärretään sen merkitys ja tavoitteet. Tämän jälkeen tarkastellaan syvällisemmin mikropalveluarkkitehtuurin hajautetun rakenteen ja palveluiden itsenäisyyden aiheuttamia haasteita, jotka vaikuttavat suoraan siihen, miten käyttäjien toimintaa voidaan seurata ja analysoida. Haasteet, joita käsitellään ovat datan hajautuminen, käyttäjäkontekstin säilyttäminen ja lokienhallinta. Kukin haaste on jaoteltu erillisiin alalukuihin, joissa kussakin käsitellään kyseinen haaste yksityiskohtaisesti.

3.1 Käyttäjien toiminnan seuranta

Käyttäjien toiminnan seuranta (engl. *user activity monitoring*) on prosessi, jossa Attererin, Wnukin ja Schmidtin (2006) mukaan kerätään ja analysoidaan tietoja käyttäjien vuorovaikutuksesta järjestelmän kanssa. Tällaisia tietoja ovat esimerkiksi käyttäjän navigointipolut, palvelussa vietetty aika ja erilaisten ominaisuuksien käyttö. Näiden tietojen avulla voidaan tunnistaa käyttäjien toimintamalleja ja mahdollisia ongelmakohtia järjestelmässä. Seurannan päätavoitteena on syventää ymmärrystä palvelun käytöstä, parantaa käyttäjäkokemusta ja ratkaista havaitut ongelmat kokonaisvaltaisesti. Tsain ym. (2018) mukaan seuranta mahdollistaa myös poikkeavan käyttäytymisen havaitsemisen, mikä saattaa viitata tietoturvauhkiin tai väärinkäytöksiin. Tällaiset poikkeamat voidaan havaita analysoimalla tavanomaisesta poikkeavia käyttötrendejä tai epätyypillistä resurssien kulutusta. Zhang ja Shafiq (2022) mainitsevat, että käyttäjien toiminnan seuranta perustuu yleensä lokitietoihin ja metriikoihin, jotka tallennetaan järjestelmän tietokantaan analysointia varten. Näitä tietoja analysoimalla voidaan tunnistaa käyttäjien toimintamalleja, kuten suosituimpia toimintoja ja käyttöaikoja.

Tsai ym. (2018) osoittavat, että analysoimalla käyttäjien vuorovaikutuksia palvelun kanssa voidaan tunnistaa käyttöliittymän ongelmakohdat, kuten monimutkaiset navigointipolut tai epäselvät toiminnot. Näin voidaan kehittää ratkaisuja, jotka parantavat palvelun käytettävyyttä ja lisäävät käyttäjätyytyväisyyttä. Zhangin ja Shafiqin (2022) mukaan jatkuva seuran-

ta paljastaa myös suorituskykyongelmia, kuten hitaat vasteajat tai palvelukatkokset. Tämä parantaa palvelun luotettavuutta, mikä on olennaista käyttäjien luottamuksen säilyttämiseksi.

Käyttäjien toiminnan seurantaan kuuluu myös tietosuojan ja tietoturvan huomioiminen. Kerättävän datan tulee olla olennaista ja tietosuojasäädösten mukaista, huomioiden tietoturva- ja tietosuojavaatimukset, kuten EU:n yleisen tietosuoja-asetuksen (GDPR) määräykset (Euroopan unionin neuvosto 2016). Seurannan on oltava läpinäkyvää ja dataa tulisi kerätä vain ilmoitettuihin tarkoituksiin. Tulee myös varmistaa, että käyttäjät ovat tietoisia siitä, mitä tietoja heistä kerätään ja mihin tarkoitukseen niitä käytetään. Lisäksi Tsain ym. (2018) mukaan on tärkeää varmistaa, että käyttäjätietoja ei vuoda palveluiden välillä ja että tiedot ovat suojattuja ulkopuolisilta uhkilta.

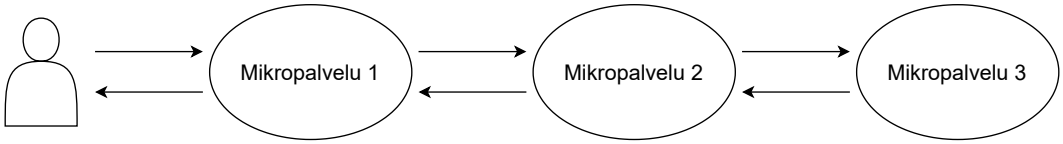
Mikropalveluarkkitehtuurissa seurannan periaatteet säilyvät, mutta niiden toteuttaminen on monimutkaisempaa järjestelmän hajautetun luonteen vuoksi. Hajautetussa ympäristössä käyttäjien toimintaan liittyvä data on hajautunut useisiin palveluihin, mikä vaikeuttaa kokonaiskuvan muodostamista ja vaatii uusia lähestymistapoja datan keräämiseen ja analysointiin. Seuraavaksi tarkastellaan käyttäjien toiminnan seurannan haasteita mikropalveluarkkitehtuurissa.

3.2 Datan hajautuminen mikropalveluissa

Yksi merkittävä haaste mikropalveluarkkitehtuurissa on datan hajautuminen useisiin eri palveluihin. Nadareishvili ym. (2016) huomauttavat, että perinteisissä monoliittisissa järjestelmissä kaikki data käsitellään keskitetysti yhdessä järjestelmässä, mikä tekee käyttäjien toiminnan seurannasta ja analysoinnista suoraviivaista. Sen sijaan mikropalveluarkkitehtuurissa jokainen palvelu hallinnoi omaa dataansa itsenäisesti, mikä Thönesin (2015) mukaan tukee palveluiden riippumattomuuden periaatetta. Tämä arkkitehtuurinen piirre tekee kokonaisvaltaisen kuvan muodostamisesta käyttäjien toiminnasta haastavampaa, sillä dataa ei ole keskitetty yhteen paikkaan. Rios, Jha ja Shwartz (2022) korostavat, että datan hajautuminen mikropalveluarkkitehtuurissa asettaa haasteita datan yhdistämiselle ja analysoinnille. Datan hajautuminen voi johtaa tilanteisiin, joissa käyttäjien toiminnan seuraaminen vaatii tietojen

yhdistämistä useista eri palveluista. Näiden hajautettujen tietojen yhdistäminen yhtenäiseksi kokonaiskuvaksi on monimutkaista ja vaatii erityisiä ratkaisuja.

Kuvio 1 havainnollistaa, miten käyttäjän pyyntö kulkee useiden itsenäisten mikropalveluiden läpi ja miten data hajautuu järjestelmässä. Kun käyttäjä lähettää pyynnön ensimmäiselle mikropalvelulle, tämä palvelu voi kutsua muita mikropalveluita suorittamaan tarvittavat toiminnot. Jokainen mikropalvelu vastaa oman osuutensa käsittelystä.



Kuvio 1. Käyttäjäpyynnön kulku ja datan hajautuminen mikropalveluarkkitehtuurissa

Käyttäjien toimintojen tapahtuminen eri palveluissa Salahin ym. (2016) mukaan vaikeuttaa käyttäjäpolkujen seuraamista. Ilman keskitettyä näkymää on haastavaa tunnistaa pullonkauloja tai virhetilanteita. Haasteen ratkaisemiseksi tarvitaan mekanismeja, jotka mahdollistavat tietojen yhdistämisen eri lähteistä, jotta käyttäjien toimintaa voidaan seurata läpi koko järjestelmän.

3.3 Käyttäjäkontekstin säilyttäminen palveluiden välillä

Käyttäjäkontekstin säilyttäminen palveluiden välillä on yksi haaste mikropalveluarkkitehtuurissa, erityisesti sen hajautetun luonteen vuoksi. Parker ym. (2020) määrittelevät käyttäjäkontekstin tiedoksi yksittäisestä käyttäjästä ja hänen toiminnastaan järjestelmässä, kuten käyttäjätunnisteeksi (engl. *user ID*). Mace ja Fonseca (2018) mainitsevat, että on tärkeää säilyttää tieto siitä, kuka käyttäjä on ja mitä hän tekee, kun käyttäjän pyyntö kulkee palveluiden läpi. Heidän mukaansa kontekstin säilyttäminen mahdollistaa käyttäjäkohtaisten toimintojen seuraamisen, mikä on tärkeää sekä käyttäjäkokemuksen että järjestelmän ylläpidon kannalta.

Macen ja Fonsecan (2018) mukaan ilman käyttäjäkontekstin välittämistä palveluiden välillä menetetään näkyvyys yksittäisten käyttäjien toimintoihin. Tämä vaikeuttaa muun muassa virheiden diagnosointia. Esimerkiksi, jos käyttäjä kohtaa palvelussa virheen, mutta käyttäjäkonteksti ei ole säilynyt, on vaikeaa tunnistaa, mille käyttäjälle virhe tapahtui.

Parker ym. (2020) toteavat, että säilyttämällä käyttäjäkonteksti voidaan virheet ja poikkeavat tapahtumat yhdistää tarkasti yksittäisiin käyttäjiin. Tämä nopeuttaa ongelmien paikantamista ja korjaamista, sillä kehittäjät voivat analysoida juuri sen käyttäjän vuorovaikutuksia, joka kohtaa toistuvia virheitä. Käyttäjäkontekstin säilyttäminen tehostaa myös järjestelmän tietoturvaa. Parkerin ym. (2020) mukaan, kun käyttäjien toimia voidaan seurata tarkasti, poikkeava tai epäilyttävä käyttäytyminen voidaan havaita nopeasti.

Käyttäjäkontekstin säilyttämiseen liittyy useita teknisiä haasteita. Macen ja Fonsecan (2018) mukaan jokaisen mikropalvelun on kyettävä vastaanottamaan, käsittelemään ja välittämään käyttäjäkonteksti eteenpäin seuraaville palveluille, mikä edellyttää standardoitujen protokollien ja käytäntöjen käyttöä. On myös varmistettava, että käyttäjäkonteksti säilyy eheänä ja luotettavana koko pyyntöketjun ajan.

3.4 Lokienhallinta hajautetussa ympäristössä

Hajautetussa mikropalveluarkkitehtuurissa jokainen palvelu toimii itsenäisesti ja tuottaa omat lokitietonsa, jotka usein tallentuvat paikallisesti palvelun hallitsemaan tietokantaan (Ahmed ym. 2020). Ahmedin ym. (2020) mukaan tämä hajautettu lokien tallennus asettaa haasteita lokienhallinnalle, sillä ilman keskitettyä järjestelmää lokitietojen yhdistäminen ja analysointi on monimutkaista ja aikaa vievää.

Lokienhallinta (engl. *log management*) tarkoittaa Kentin ja Souppayan (2006) mukaan järjestelmän tuottamien aikajärjestyksessä kirjattujen merkintöjen, eli lokitietojen, keräämistä, tallentamista, käsittelyä ja analysointia. Heidän mukaansa lokitiedot sisältävät usein tietoa käyttäjien toimenpiteistä, järjestelmän sisäisistä prosesseista sekä palvelun suorituskykyyn ja toimintaan liittyvistä tapahtumista. Näiden tietojen avulla järjestelmän ylläpitäjät ja kehittäjät voivat seurata sovelluksen toimintaa, diagnosoida mahdollisia ongelmia sekä parantaa palvelun luotettavuutta ja käytettävyyttä.

Monoliittisissa järjestelmissä lokitietojen kokoaminen yhteen on suhteellisen helppoa, koska kaikki komponentit toimivat samassa ympäristössä, jolloin monimutkaisia yhdistämismekanismeja ei tarvita. Mikropalveluympäristössä tämä ei välttämättä ole mahdollista ilman erityisiä toimenpiteitä, sillä Nadareishvili ym. (2016) huomauttavat, että palvelut voivat sijaita

eri palvelimilla. Lisäksi palvelut voivat olla toteutettu eri teknologioilla ja käyttää erilaisia lokitusformaatteja, mikä vaikeuttaa lokitietojen yhdistämistä ja analysointia.

Boncea, Zamfiroiu ja Bacivarov (2018) mainitsevat lokitietojen keräämisen useista eri lähteistä vaativan mekanismeja, jotka pystyvät kommunikoimaan eri palveluiden välillä ja siirtämään dataa tehokkaasti. Jotta lokitietoja voidaan analysoida keskitetysti, ne on yhdistettävä yhteen järjestelmään, mikä edellyttää standardointia lokitietojen rakenteessa.

4 Ratkaisut käyttäjien toiminnan seurantaan

Tässä luvussa käsitellään keskeisiä ratkaisuja, joilla voidaan tehokkaasti toteuttaa käyttäjien toiminnan seuranta mikropalveluarkkitehtuurissa. Keskitytään hajautettuun jäljitykseen, käyttäjäkontekstin välittämiseen palveluiden välillä sekä keskitettyyn lokienhallintaan. Näiden menetelmien avulla voidaan vastata aiemmin esiin tuotuihin haasteisiin ja mahdollistaa kokonaisvaltainen käyttäjien toiminnan seuranta hajautetuissa järjestelmissä. Lopuksi esitellään kokonaisvaltainen käyttäjien toiminnan seurantamalli, joka yhdistää nämä eri ratkaisut yhtenäiseksi kokonaisuudeksi.

4.1 Hajautettu jäljitys

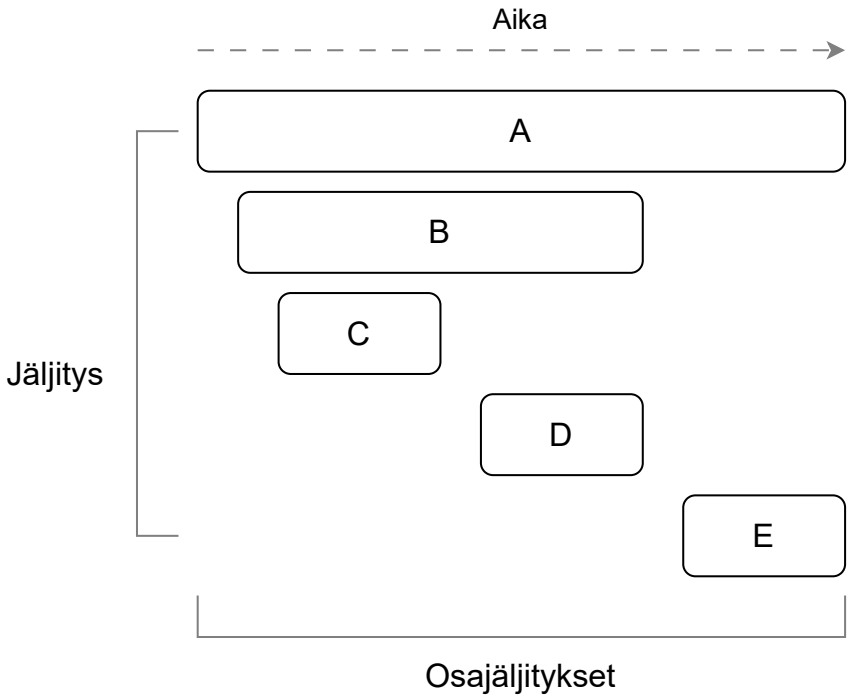
Hajautettu jäljitys (engl. *distributed tracing*) on tekniikka, joka Shkuron (2019) mukaan mahdollistaa käyttäjien pyyntöjen seurannan hajautetuissa järjestelmissä, kuten mikropalveluarkkitehtuurissa. Se tarjoaa näkyvyyttä järjestelmän sisäiseen toimintaan, auttaen ymmärtämään, miten yksittäiset palvelut käsittelevät pyyntöjä ja miten ne vaikuttavat toisiinsa. Tämä on erityisen tärkeää virheiden diagnosoinnissa ja suorituskykyongelmien tunnistamisessa.

Parkerin ym. (2020) mukaan, kun tapahtumat voidaan yhdistää kokonaisuudeksi, järjestelmät pystyvät tarjoamaan entistä yksityiskohtaisempaa seurantaa ja analytiikkaa. Tämä tarkoittaa, että palvelun käytöstä ja suorituskyvystä saadaan entistä tarkempaa ja yksityiskohtaisempaa dataa. Shkuron (2019) mukaan tällainen data on arvokasta palvelun optimoinnissa, sillä sen avulla voidaan tunnistaa esimerkiksi suosituimmat ominaisuudet ja käyttömäärien huiput. Tämä myös mahdollistaa resurssien tehokkaamman kohdentamisen ja palvelun laadun parantamisen käyttäjien tarpeiden mukaisesti.

Hajautetun jäljityksen keskeinen osa on hyödyntää uniikkeja tunnisteita käyttäjien toiminnan seuraamiseksi palvelusta toiseen. Nämä tunnisteet jakautuvat kahteen päätyyppiin (Shkuro 2019): jäljitystunniste (engl. *Trace ID*) ja osajäljitystunniste (engl. *Span ID*). Jäljitystunniste kuvaa koko pyyntöketjun alusta loppuun mahdollistaen yhtenäisen seurannan palveluiden välillä. Osajäljitystunniste taas kuvaa yksittäisen palvelun suorittamaa operaatiota tarjoten

yksityiskohtaisen näkymän pyyntöketjun rakenteeseen. Kuten Shkuro (2019) selittää, hajautetussa jäljityksessä jokaiselle käyttäjän pyynnölle luodaan ainutlaatuinen jäljitystunniste, joka kulkee palvelusta toiseen pyyntöketjun mukana. Palvelut lisäävät omat jäljitystietonsa, jotka kuvaavat yksittäisiä operaatioita osana kokonaisjälkeä, mikä mahdollistaa täydellisen pyyntöpolun rekonstruoinnin.

Kuviossa 2 havainnollistetaan hajautetun jäljityksen toimintaa mikropalveluarkkitehtuurissa. Kuvion aikajana osoittaa tapahtumien etenemisen ajan suhteen vasemmalta oikealle. Jokainen palkki (A, B, C, D ja E) edustaa yksittäistä operaatiota. Näiden operaatioiden muodostama kokonaisuus edustaa yhtä jäljitystä. Kuvion avulla voidaan hahmottaa, miten pyyntö etenee eri mikropalveluiden välillä ja kuinka eri toimintojen kestot vaihtelevat.



Kuvio 2. Hajautettu jäljitys mikropalveluarkkitehtuurissa (Jaeger 2024)

Jäljitystunnisteiden hallinta edellyttää, että palvelut kykenevät vastaanottamaan, käsittelemään ja välittämään nämä tunnisteet luotettavasti. Riosin, Jhan ja Shwartzin (2022) mukaan tämä edellyttää palveluiden instrumentointia sekä yhteisten protokollien ja standardien käyt-

töä jäljitystietojen käsittelyssä ja välittämisessä, jotta tiedot ovat yhteensopivia ja yhdistettävissä. Mace ja Fonseca (2018) määrittelevät palveluiden instrumentoinnin toiminnoksi, jossa palveluita muokataan siten, että ne voivat kerätä ja välittää jäljitystietoja tehokkaasti. Parker ym. (2020) toteavat hajautetun jäljityksen implementoinnin olevan haastavaa eri teknologioita käyttävissä palveluissa. Ratkaisuna tähän haasteeseen on palveluiden instrumentointi yhteisten instrumentointikirjastojen avulla, jotka mahdollistavat jäljitystietojen keräämisen yhtenäisellä tavalla.

Jäljitysdatan kerääminen ja analysointi vaatii tehokkaita työkaluja. Rios, Jha ja Shwartz (2022) mainitsevat Jaegerin olevan yksi avoimen lähdekoodin työkaluista hajautetun jäljityksen toteuttamiseen. Jaeger tukee OpenTelemetry-standardia, mikä mahdollistaa yhteensopivuuden eri järjestelmien välillä (Jaeger 2024). On olemassa myös muita vastaavia työkaluja, kuten Zipkin (Zipkin 2024), mutta koska niiden toimintaperiaate on hyvin samankaltainen kuin Jaegerin, keskitytään tässä tutkimuksessa ainoastaan Jaegeriin.

Jaeger on rakennettu skaalautuvaksi ja hajautetuksi järjestelmäksi, joka vastaanottaa ja tallentaa palveluiden lähettämät jäljitystiedot keskitettyyn tietokantaan (Parker ym. 2020). Parkerin ym. (2020) mukaan näitä tietoja voidaan visualisoida ja analysoida Jaegerin käyttöliittymän kautta, mikä helpottaa järjestelmän suorituskyvyn optimointia ja virheiden diagnosointia. Koska Jaeger tukee standardoituja protokollia, se soveltuu hyvin toimimaan eri teknologioilla toteutettujen palveluiden kanssa ja mahdollistaa hajautetun jäljityksen toteuttamisen yhtenäisesti koko järjestelmässä.

OpenTelemetry on avoin standardi, joka määrittelee yhteiset rajapinnat ja käytännöt jäljitystietojen keräämiseksi, käsittelemiseksi ja välittämiseksi hajautetuissa järjestelmissä (OpenTelemetry 2024). Se tarjoaa yhtenäisen lähestymistavan telemetriatietojen hallintaan. Parker ym. (2020) määrittelevät telemetriatietojen viittaavan järjestelmän toiminnasta kerättyyn metriikoihin, kuten suorituskykyyn ja käyttöasteisiin joiden avulla voidaan analysoida järjestelmän toimintaa. Heidän mukaansa OpenTelemetryn avulla kehittäjät voivat helposti instrumentoida ohjelmistojaan, kerätä jäljitystietoja ja metriikoita sekä lähettää ne keskitettyihin analytiikkajärjestelmiin kuten Jaegeriin.

4.2 Käyttäjäkontekstin välittäminen

Käyttäjäkontekstin välittäminen (engl. *user context propagation*) palveluiden välillä on keskeinen osa tehokasta käyttäjien toiminnan seurantaa mikropalveluarkkitehtuurissa. Kuten Parker ym. (2020) huomauttavat, ilman asianmukaista kontekstin siirtämistä jokaisen palvelun läpi on haastavaa muodostaa yhtenäistä näkymää käyttäjän toiminnasta.

Shkuro (2019) toteaa, että käyttäjäkontekstin välittäminen edellyttää, että jokainen palvelu osaa vastaanottaa ja välittää käyttäjäkontekstin eteenpäin seuraaville palveluille. Tämä tarkoittaa esimerkiksi käyttäjätunnisteen tai muiden käyttäjää kuvaavien tietojen liittämistä palveluiden välisiin pyyntöihin. Parkerin ym. (2020) mukaan yleinen tapa välittää käyttäjäkontekstia on lisätä tarvittavat tiedot palveluiden välisiin viesteihin HTTP-otsikoiden kautta. Tällöin jokainen palvelu pyyntöketjussa kykenee tunnistamaan, kuka käyttäjä on kyseessä.

Onnistuneen käyttäjäkontekstin välittämisen ansiosta järjestelmä kykenee erittelemään käyttäjäkohtaisia ongelmia. Parkerin ym. (2020) mukaan kehittäjät voivat tarpeen tullen analysoida yksittäisen käyttäjän vuorovaikutusta järjestelmässä, tunnistaa toistuvat virheet tai havaitsemaan esimerkiksi poikkeavan toiminnan, joka saattaisi viitata tietoturvauhkiin. Näin käyttäjäkontekstin välittäminen paitsi nopeuttaa ongelmien paikantamista ja ratkaisemista, myös parantaa järjestelmän tietoturvaa ja käyttäjäkokemusta.

4.3 Keskitetty lokienhallinta

Keskitetty lokienhallinta (engl. *centralized log management*) on olennainen osa käyttäjien toiminnan seurantaa mikropalveluarkkitehtuurissa ja tarjoaa ratkaisun hajautetun ympäristön aiheuttamiin haasteisiin lokitietojen keräämisessä, tallentamisessa ja analysoinnissa. Talaşin, Popin ja Neagin (2017) mukaan se mahdollistaa kaikkien mikropalveluiden tuottamien lokien tehokkaan keräämisen yhteen paikkaan, mikä parantaa järjestelmän hallintaa ja valvontaa. Sisällyttämällä lokitietoihin olennainen metadata, kuten aikaleimat, palvelun nimet ja jäljitystunnisteet, voidaan helpottaa lokitietojen suodattamista ja yhdistämistä.

Keskitetty lokienhallinta tarjoaa merkittäviä etuja, joiden avulla voidaan parantaa sekä järjestelmän suorituskykyä että hallittavuutta mikropalveluarkkitehtuurissa. Boncea, Zamfiroiu

ja Bacivarov (2018) korostavat, että se mahdollistaa kokonaisvaltaisen näkyvyyden järjestelmän toimintaan keräämällä kaikkien palveluiden lokitiedot yhteen paikkaan. Tämä keskitetty lähestymistapa helpottaa käyttäjien toiminnan seurantaa, sillä kaikki lokitiedot ovat saatavilla yhdestä paikasta, mikä nopeuttaa niiden analysointia.

Keskitetyt lokienhallintajärjestelmät on suunniteltu skaalautuviksi, eli ne pystyvät käsittelemään suuria määriä dataa ja mukautumaan järjestelmän kasvaessa, kuten Chhajed (2015) korostaa. Tämä ominaisuus on erityisen tärkeä mikropalveluarkkitehtuurissa, jossa palveluiden määrä voi kasvaa merkittävästi. Skaalautuva lokienhallinta varmistaa, että järjestelmä pysyy tehokkaana ja luotettavana myös kuormituksen kasvaessa.

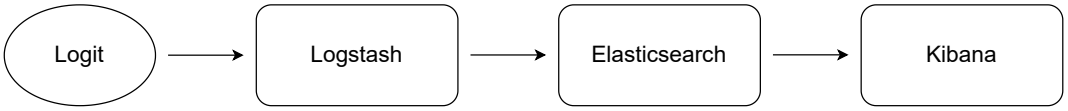
Keskitetty lokienhallinta tukee tehokasta vianmääritystä yhdistämällä eri palveluiden lokitiedot, mikä mahdollistaa virheiden nopean tunnistamisen ja korjaamisen, kuten Boncea, Zamfiroiu ja Bacivarov (2018) painottavat. Heidän mukaansa keskitettyjen lokienhallintajärjestelmien avulla voidaan suorittaa kehittynyttä analytiikkaa ja raportointia. Näiden tietojen pohjalta voidaan tehdä perusteltuja parannuksia palveluun ja vahvistaa sen tietoturvaa.

Talaş, Pop ja Neagu (2017) esittävät Elastic Stackin yhtenä ratkaisuna keskitetyn lokienhallinnan toteuttamiseen keräämään lokitiedot yhteen paikkaan keskitysti. Heidän mukaan Elastic Stack on avoimen lähdekoodin alusta, joka koostuu Elasticsearchista, Logstashista ja Kibanasta. Vaikka on olemassa myös muita vastaavia työkaluja, kuten Graylog (Graylog 2024), keskitytään tässä tutkimuksessa vain Elastic Stackiin, koska ne toimivat pääosin samalla periaatteella.

Elastic Stackin keskeiset komponentit ovat (Talaş, Pop ja Neagu 2017):

- **Elasticsearch**: Hajautettu hakukone ja analytiikkajärjestelmä, joka tallentaa ja indeksoi lokitiedot. Elasticsearch tarjoaa rajapinnan, jonka avulla lokitietoja voidaan hakea.
- **Logstash**: Datankäsittelyputki, joka kerää, muokkaa ja reitittää lokitiedot eri lähteistä Elasticsearchiin. Logstash tukee useita eri syötteitä ja mahdollistaa lokitietojen keräämisen monista eri lähteistä.
- **Kibana**: Visualisointityökalu, joka tarjoaa käyttöliittymän lokitietojen analysointiin ja visualisointiin. Kibana mahdollistaa interaktiivisten kaavioiden ja taulukoiden luomisen, joiden avulla voidaan tarkastella ja analysoida lokitietoja.

Kuviossa 3 esitetään Elastic Stackin arkkitehtuuri. Tämän avulla voidaan toteuttaa tehokas ja skaalautuva lokienhallintajärjestelmä, joka soveltuu hyvin mikropalveluympäristöihin.



Kuvio 3. Elastic Stackin arkkitehtuuri (Talaş, Pop ja Neagu 2017)

Elastic Stack voidaan ottaa käyttöön itse ylläpidetyssä ympäristössä rakentamalla oma keskitetty lokienhallintajärjestelmä. Toinen vaihtoehto on käyttää pilvipohjaisia palveluita, kuten AWS Elasticsearch Serviceä, Azure Monitoria tai Google Cloudin Stackdriveria (Elastic 2024). Nämä palvelut tarjoavat keskitetyn lokienhallinnan ilman tarvetta ylläpitää omaa järjestelmää.

4.4 Kokonaisvaltainen käyttäjien toiminnan seurantamalli

Kokonaisvaltainen käyttäjien toiminnan seurantamalli mikropalveluarkkitehtuurissa yhdistää hajautetun jäljityksen, käyttäjäkontekstin välittämisen ja keskitetyn lokienhallinnan yhdeksi tehokkaaksi ratkaisuksi. Tämä mahdollistaa järjestelmän tarkastelun sekä yksittäisten palveluiden että koko arkkitehtuurin tasolla.

Mallin ensimmäinen osa on hajautettu jäljitys OpenTelemetryn ja Jaegerin avulla, mikä mahdollistaa pyyntöketjujen seurannan läpi koko mikropalveluarkkitehtuurin (Parker ym. 2020). OpenTelemetry tarjoaa yhtenäisen tavan kerätä jäljitystietoja eri palveluista, ja Jaeger vastaanottaa sekä tallentaa nämä tiedot analysointia varten.

Toinen osa on käyttäjäkontekstin välittäminen palveluiden välillä. Liittämällä käyttäjäkonteksti jäljitystietoihin voidaan yhdistää yksittäiset käyttäjien toiminnot koko järjestelmän tasolla (Parker ym. 2020). Tämä mahdollistaa käyttäjäkohtaisten ongelmien tunnistamisen ja käyttäjäkokemuksen parantamisen.

Kolmas osa mallia on keskitetty lokienhallinta Elastic Stackin avulla. Elastic Stack tarjoaa kattavan näkyvyyden järjestelmän toimintaan keräämällä kaikki palveluiden tuottamat lokitiedot yhteen paikkaan (Talaş, Pop ja Neagu 2017). Keskitetty lokienhallinta mahdollistaa

tehokkaan vianmäärityksen ja suorituskyvyn optimoinnin sekä tukee kehittynyttä analytiikkaa ja raportointia.

Yhdistämällä nämä kolme osaa muodostuu tehokas seurantamalli, joka vastaa mikropalveluarkkitehtuurin asettamiin haasteisiin. Hajautettu jäljitys mahdollistaa pyyntöjen seuraamisen palvelusta toiseen, käyttäjäkontekstin välittäminen yhdistää toiminnot yksittäisiin käyttäjiin, ja keskitetty lokienhallinta tarjoaa alustan lokitietojen tallentamiseen ja analysointiin. Tämä kokonaisvaltainen malli parantaa järjestelmän hallittavuutta, suorituskykyä ja tietoturvaa sekä mahdollistaa paremman käyttäjäkokemuksen tarjoamisen.

Oletetaan mikropalveluarkkitehtuuri, jossa on useita palveluita. Jokainen palvelu on instrumentoitu OpenTelemetryllä. Kun käyttäjä suorittaa toiminnon, joka kulkee useiden palveluiden läpi, OpenTelemetry kerää tämän toiminnon aikana syntyvät jäljitystiedot. Nämä tiedot lähetetään Jaegeriin, ja niihin liitetään myös käyttäjäkonteksti, kuten käyttäjätunnus. Näin tapahtumat voidaan yhdistää yksittäiseen käyttäjään ja seurata hänen toimintaansa järjestelmässä kokonaisvaltaisesti. Samanaikaisesti palveluiden tuottamat lokitiedot kerätään Logstashin kautta Elasticsearchiin, missä niitä voidaan tarkastella ja analysoida Kibanalla. Näin muodostuu kokonaisvaltainen käyttäjien toiminnan seurantamalli, joka tarjoaa syvällisen näkyvyyden käyttäjien toimintaan.

5 Yhteenveto

Tämän tutkimuksen tavoitteena on ollut selvittää, kuinka käyttäjien toiminnan seurantaa voidaan toteuttaa tehokkaasti hajautetussa mikropalveluarkkitehtuurissa, jossa käyttäjädata ja tapahtumatiedot jakautuvat eri palveluiden kesken. Mikropalveluarkkitehtuurin tarjoama joustavuus ja skaalautuvuus tekevät siitä suositun ratkaisun modernissa ohjelmistokehityksessä, mutta samalla se on tuonut merkittäviä haasteita erityisesti käyttäjien toiminnan seurannalle.

Keskeisiä haasteita ovat olleet datan hajautuminen, käyttäjäkontekstin säilyttäminen palveluiden välillä ja lokienhallinta hajautetussa ympäristössä. Datan hajautuminen vaikeuttaa kokonaisvaltaisen näkymän saamista käyttäjien toiminnasta, koska tiedot ovat jakautuneet useisiin itsenäisiin palveluihin. Käyttäjäkontekstin säilyttäminen on osoittautunut kriittiseksi, jotta yksittäiset käyttäjien toiminnot voidaan yhdistää kokonaiskuvaksi heidän toiminnastaan järjestelmässä. Lokienhallinta on puolestaan haastavaa ilman keskitettyä järjestelmää, joka mahdollistaa lokitietojen tehokkaan keräämisen ja analysoinnin.

Tässä tutkimuksessa on osoitettu, että hajautettu jäljitys, käyttäjäkontekstin välittäminen ja keskitetty lokienhallinta tarjoavat yhdessä kokonaisvaltaisen seurantamallin, joka vastaa näihin haasteisiin. Hajautettu jäljitys mahdollistaa pyyntöjen seuraamisen mikropalveluarkkitehtuurin läpi. Käyttäjäkontekstin tehokas välittäminen varmistaa, että käyttäjään liittyvä tieto säilyy ehjänä koko pyyntöketjun ajan. Keskitetty lokienhallinta yhdistää hajautetun datan yhdeksi analysoitavaksi kokonaisuudeksi, mahdollistaen reaaliaikaisen seurannan.

Käyttäjien toiminnan seuranta mikropalveluarkkitehtuurissa on osoittautunut monimutkaiseksi haasteeksi johtuen järjestelmän hajautetusta luonteesta. Esitetyt ratkaisut kuitenkin osoittavat, että haasteet ovat voitettavissa oikeilla ratkaisuilla. Kokonaisvaltainen käyttäjien toiminnan seurantamalli ei ole pelkästään tekninen edellytys, vaan se muodostaa perustan korkealaatuisille, luotettaville ja käyttäjälähtöisille palveluille. Tulevaisuudessa mikropalveluarkkitehtuuri tulee todennäköisesti yleistymään entisestään, mikä korostaa tutkimuksen merkitystä.

Jatkossa tutkimusta voitaisiin laajentaa tekoälyn ja koneoppimisen hyödyntämiseen kerätyn

datan analysoinnissa, mikä avaisi uusia mahdollisuuksia järjestelmän optimointiin ja ennakoivaan ylläpitoon. Toinen potentiaalinen jatkotutkimuksen kohde olisi käyttäjien toiminnan seurannan hyödyntäminen käyttäjäkokemuksen parantamiseksi. Tällöin voitaisiin kehittää entistä käyttäjälähtöisempiä palveluita, jotka vastaavat paremmin käyttäjien tarpeisiin.

Lähteet

Ahmed, Farrukh, Urooj Jahangir, Hamad Rahim, Kamran Ali ym. 2020. “Centralized log management using elasticsearch, logstash and kibana”. Teoksessa *2020 International Conference on Information Science and Communication Technology (ICISCT),* 1–7. IEEE. https://doi.org/10.1109/ICISCT49550.2020.9080053.

Alpers, Sascha, Christoph Becker, Andreas Oberweis ja Thomas Schuster. 2015. “Microservice Based Tool Support for Business Process Modelling”. Teoksessa *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop,* 71–78. https://doi.org/10.1109/EDOCW.2015.32.

Atterer, Richard, Monika Wnuk ja Albrecht Schmidt. 2006. “Knowing the user’s every move: user activity tracking for website usability evaluation and implicit interaction”. Teoksessa *Proceedings of the 15th international conference on World Wide Web,* 203–212. ACM. https://doi.org/10.1145/1135777.1135811.

Boncea, Radu, Alin Zamfiroiu ja Ioan Bacivarov. 2018. “A scalable architecture for automated monitoring of microservices”. *Academy of Economic Studies. Economy Informatics* 18 (1): 13–22.

Chhajed, Saurabh. 2015. *Learning ELK stack.* Packt Publishing Ltd.

Dragoni, Nicola, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin ja Larisa Safina. 2017. “Microservices: yesterday, today, and tomorrow”. *Present and ulterior software engineering,* 195–216. https://doi.org/10.1007/978-3-319-67425-4_12.

Elastic. 2024. https://www.elastic.co. Viitattu: 27.9.2024.

Euroopan unionin neuvosto. 2016. *General Data Protection Regulation 2016/679.* Https://eur-lex.europa.eu/eli/reg/2016/679/oj.

Graylog. 2024. https://graylog.org. Viitattu: 17.11.2024.

Jaeger. 2024. https://www.jaegertracing.io. Viitattu: 2.10.2024.

Kent, Karen Ann ja Murugiah Souppaya. 2006. “Guide to Computer Security Log Management”, https://doi.org/10.6028/NIST.SP.800-92.

Mace, Jonathan ja Rodrigo Fonseca. 2018. “Universal context propagation for distributed system instrumentation”. Teoksessa *Proceedings of the thirteenth EuroSys conference,* 1–18. ACM. https://doi.org/10.1145/3190508.3190526.

Nadareishvili, Irakli, Ronnie Mitra, Matt McLarty ja Mike Amundsen. 2016. *Microservice architecture: aligning principles, practices, and culture.* "O’Reilly Media, Inc."

Newman, Sam. 2021. *Building microservices.* "O’Reilly Media, Inc."

OpenTelemetry. 2024. https://opentelemetry.io. Viitattu 8.10.2024.

Parker, Austin, Daniel Spoonhower, Jonathan Mace, Ben Sigelman ja Rebecca Isaacs. 2020. *Distributed tracing in practice: Instrumenting, analyzing, and debugging microservices.* O’Reilly Media.

Rios, Jesus, Saurabh Jha ja Laura Shwartz. 2022. “Localizing and explaining faults in microservices using distributed tracing”. Teoksessa *2022 IEEE 15th International Conference on Cloud Computing (CLOUD),* 489–499. IEEE. https://doi.org/10.1109/CLOUD55607.2022.00072.

Salah, Tasneem, M Jamal Zemerly, Chan Yeob Yeun, Mahmoud Al-Qutayri ja Yousof Al-Hammadi. 2016. “The evolution of distributed systems towards microservices architecture”. Teoksessa *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST),* 318–325. IEEE. https://doi.org/10.1109/ICITST.2016.7856721.

Shkuro, Yuri. 2019. *Mastering Distributed Tracing: Analyzing performance in microservices and complex systems.* Packt Publishing Ltd.

Talaş, Andrei, Florin Pop ja Gabriel Neagu. 2017. “Elastic stack in action for smart cities: Making sense of big data”. Teoksessa *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP),* 469–476. IEEE. https://doi.org/10.1109/ICCP.2017.8117049.

Thönes, Johannes. 2015. “Microservices”. *IEEE Software* 32 (1): 116–116. https://doi.org/10.1109/MS.2015.11.

Tsai, Pang-Wei, Chun-Wei Tsai, Chia-Wei Hsu ja Chu-Sing Yang. 2018. “Network monitoring in software-defined networking: A review”. *IEEE Systems Journal* 12 (4): 3958–3969. https://doi.org/10.1109/JSYST.2018.2798060.

Villamizar, Mario, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Ruby Casallas ja Santiago Gil. 2015. “Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud”. Teoksessa *2015 10th computing colombian conference (10ccc),* 583–590. IEEE. https://doi.org/10.1109/ColumbianCC.2015.7333476.

Zhang, Cathy H ja M Omair Shafiq. 2022. “A Real-time, Scalable Monitoring and User Analytics Solution for Microservices-based Software Applications”. Teoksessa *2022 IEEE International Conference on Big Data (Big Data),* 6125–6134. IEEE. https://doi.org/10.1109/BigData55660.2022.10020831.

Zipkin. 2024. https://zipkin.io. Viitattu: 2.10.2024.