Tampereen yliopisto

Otto

SUORITUSKYVYNTESTAUSTYÖKALUN VALINTA WEB-SOVELLUSPALVELUIDEN TESTAAMISEEN

Kandidaatintutkielma
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Petri
Toukokuu 2023

TIIVISTELMÄ

Otto: Suorituskyvyntestaustyökalun valinta web-sovelluspalveluiden testaamiseen
Kandidaatintutkielma
Tampereen yliopisto
Tieto- ja sähkötekniikan tutkinto-ohjelma
Toukokuu 2023

Suorituskyvyntestaustyökalut (engl. performance testing tool) ovat testiautomaatioon liittyviä työkaluja, joilla voidaan testata järjestelmän suorituskykyä. Suorituskyvyntestaustyökaluja on monia erilaisia ja niistä pitää osata valita oikea omiin tarpeisiin. Erilaisia ohjelmistoja, joilla on tarvetta suorituskyvyntestaukseen, on erittäin monia. Tästä syytä tämän työn aihe on rajattu web-sovelluspalveluiden suorituskyvyntestaamiseen. Miten valitaan oikea suorituskyvyntestaustyökalu web-sovelluspalveluiden testaamiseen? Mitä tekijöitä pitää ottaa huomioon, kun vertaillaan eri työkaluja?

Web-sovelluspalvelut (engl. web service) ovat ohjelmistojärjestelmiä, jotka mahdollistavat tietokoneiden keskeisen vuorovaikutuksen. Toisien sanottuna ne ovat hyvin keskeinen osa nykymaailman IT-infrastruktuuria. Tästä syystä niiden moitteettomaan toimimiseen pyritään ohjelmistotestauksella, jonka yksi tärkeä vaihe on suorituskyvyntestaus. Suorituskyky kertoo ohjelman kyvystä toimia nopeasti ajan suhteen ja sen kyvystä käyttää saatavilla olevia resursseja tehokkaasti. Suorituskykyä testataan erikoistuneilla suorituskyvyntestaustyökaluilla.

Työ suoritettiin kirjallisuuskatsauksena, joten käytettävä aineisto saatiin suorittamalla hakuja eri hakupalveluilla. Tehdyt tiedonhaut voidaan jakaa kahteen kategoriaan. Ensimmäiseen kategoriaan kuuluu haut, joilla etsittiin suoraan vastauksia tämän työn tutkimuskysymyksiin. Toinen kategoria sisältää haut, joilla perusteltiin työn taustatietoja, tai löydettyjen suorituskyvyntestaustyökalujen valintaan liittyvien tekijöiden tärkeyttä. Kirjallisuuskatsauksessa suosittiin tieteellisiä lähteitä ja pyrittiin löytämään mielipiteiden sijaan tieteellisiä perusteluja.

Työn tulokset muodostuvat kirjallisuuskatsauksen avulla löydettyjen valintatekijöiden käsittelystä. Eri valintatekijät jaetaan selkeyden vuoksi kahteen kategoriaan, organisatorisiin tekijöihin ja työkalukohtaisiin tekijöihin. Nämä kaksi kategoriaa jaetaan molemmat vielä kolmeen erikseen käsiteltävään alakategoriaan. Organisatorisissa tekijöissä käsitellään suorituskyvyntestaustyökalujen rahallisia kustannuksia, käyttötarpeen suuruutta ja saatavilla olevaa tukea. Työkalukohtaisissa tekijöissä perehdytään tarvittavaan testaustyyppiin, yhteensopivuuteen ja käytettävyyteen. Edellä mainittuja tieteellisistä lähteistä löydettyjä valintatekijöitä vertaillaan internetistä löytyviin alan ammattilaisten henkilökohtaisiin mielipiteisiin tai oppaisiin. Vertailun perusteella saadaan selville, että tämän työn tulokset ovat kattavat ja käsittelevät suorituskyvyntestaustyökalujen valintaan liittyviä tekijöitä laajasti.

Avainsanat: suorituskyky, suorituskyvyn testaus, suorituskyvyntestaustyökalu, suorituskyvyntestaustyökalun valinta, web-sovelluspalvelu

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

API	Application Programming Interface
CI/CD	Continuous Integration and Continuous Delivery
CoAP	Constrained Application Protocol
DevOps	Software Development and IT Operations
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
IT	Information Technology
JVM	Java virtual machine
MQTT	Message Queuing Telemetry Transport
REST	Representational state transfer
SaaS	Software as a Service
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
Stack Overflow	Ohjelmointiaiheinen kysymys–vastaus-sivusto
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
W3C	World Wide Web Consortium
web-sovelluspalvelu	engl. web service, arkikielessä web-palvelu
XML	Extensible Markup Language

1. JOHDANTO

Yhä digitalisoituvassa nykymaailmassa tietoteknologiatuotteiden ja -palvelujen ala on biljoonien eurojen arvoinen. Hyvän IT-infrastruktuurin ylläpitäminen on välttämätöntä niin yrityksille, valtion palveluille kuin voittoa tavoittelemattomille järjestöillekin. (Sava, 2022) Eri IT-järjestelmiä yhdistää web-sovelluspalvut (engl. web service), joten niiden moitteeton toimiminen on kriittinen osa IT-infrastruktuuria. Web-sovelluspalveluja kehitettäessä niiden moitteettomaan toimimiseen pyritään ohjelmistotestauksen avulla. Suorituskyvyntestaus on tärkeä osa ohjelmistotestausta, jolla varmistetaan muun muassa ohjelman nopea ja tehokas toimiminen. Suorituskykyä testataan suorituskyvyntestaustyökaluilla, joita on olemassa hyvin monia erilaisia. Kuitenkin sovelluskehitysprojektia varten täytyy ostata valita tarpeisiin sopiva suorituskyvyntestaustyökalu.

Tässä työssä perehdytään tekijöihin, jotka vaikuttavat suorituskyvyntestaustyökalun valintaan. Suorituskyky, ja sen testaus, on kokonaisuudessaan todella laaja aihealue. Tämän kaiken käsittely ei mahtuisi tähän työhön. Tästä syystä työn näkökulma on rajattu web-sovelluspalveluiden suorituskyvyn testaamiseen työkaluilla. Tämä työ vastaa kysymyksiin, miten valitaan oikea suorituskyvyntestaustyökalu web-sovelluspalveluiden testaamiseen ja mitä tekijöitä pitää ottaa huomioon, kun vertaillaan eri työkaluja. Aiheesta löytyy materiaalia enimmäkseen oppaina. Oppaissa harvoin on mitään lähteitä ja ne perustuvat vain alan ihmisten omiin kokemuksiin ja mielipiteisiin. Tästä syystä on mielenkiintoista suorittaa aiheesta kirjallisuuskatsaus ja löytää tieteellisiä perusteita eri tekijöille, jotka vaikuttavat suorituskyvyntestaustyökalun valintaan.

Luvussa 2 kuvataan tutkimusmenetelmä ja tiedonhakuprosessi. Tiedonhaunprosessista kerrotaan käytetyt hakusanat ja millä perusteella lähteitä otettiin mukaan työhön. Luvussa 3 perehdytään työn aiheen taustatietoihin. Taustatietoihin kuuluu keskeisten käsitteiden ymmärtäminen, suorituskyvyntestauksen tärkeyden sekä toimintaperiaatteen hahmottaminen ja suorituskyvyntestaustyökalujen toimintaperiaatteiden käsittäminen. Luvussa 4 esitetään työn tulokset. Tulokset ovat suorituskyvyntestaustyökalujen valintaan liittyvien tekijöiden muodossa. Monia eri tekijöitä kuvaillaan esimerkkien avulla ja niiden tärkeyttä perustellaan kirjallisuuskatsauksella löydettyjen lähteiden avulla. Luvussa 5 tarkastellaan saatuja tuloksia ja vertaillaan niitä internetistä löytyviin oppaisiin. Tarkastelussa esitetään kritiikkiä tuloksia kohtaan ja pohditaan työssä olleita rajoitteita. Lopuksi luvussa 6 vedetään yhteen työn aihe ja saadut tulokset.

2. TUTKIMUSMENETELMÄ

Tämä työn on toteutettu kirjallisuuskatsauksena ja haetut lähteet voidaan jakaa karkeasti kahteen kategoriaan. Ensimmäiseksi haettiin lähteitä, jotka liittyvät suoraan tämän työn tutkimuskysymyksiin. Esimerkiksi lähteitä, joissa käsitellään suoraan tekijöitä, jotka vaikuttavat suorituskyvyntestaustyökalun valintaan. Toiseksi haettiin lähteitä, joilla perusteltiin työn taustatiedot tai tiettyjen työkalun valintaan liittyvien tekijöiden tärkeys. Esimerkiksi lähteitä, jotka kertovat mikä on web-sovelluspalvelun määritelmä, tai miksi visualisoinnit ovat tärkeitä käytettävyyden kannalta. Ensimmäiseen kategoriaan kuuluvia lähteitä etsittiin Andor ja Google Scholar -hakupalveluiden avulla. Toiseen kategoriaan kuuluvia termien määrittelyjä sisältäviä lähteitä etsittiin myös suoraan Google-haulla.

Suoraan tutkimuskysymyksiin liittyviä lähteitä etsittäessä hakusanat muodostettiin suoraan aiheen sanastosta, kuten esimerkiksi ”performance testing tool“. Taustatietojen tai tekijöiden tärkeyttä perustelevia lähteitä etsittäessä hakusanat muodostettiin, kun oli saatu selville mitä halutaan perustella. Esimerkiksi aiemmin löydetyissä tutkimustuloksissa todettiin, että käyttöliittymän käytettävyys on tärkeää, joten perustelun löytämiseksi hakusanaksi otettiin ”interface usability“. Tämän työn tiedonhakuprosessi on toistettavissa seuraavilla hakulausekkeilla:

- ”performance testing tool“ OR ”load testing tool“
- ”performance testing tools“
- performance testing web
- performance testing web service
- ”agile testing“
- interface usability
- SOAP vs REST
- MQTT vs CoAP

Melkein joka hakua suoritettaessa käytettiin Andor-hakupalvelun toimintoa rajata tulokset vain vertaisarvioituihin julkaisuihin. Hakuja suoritettaessa saatiin vaihtelevia määriä hakutuloksia kymmenistä tuhansiin. Julkaisuvuotta ei rajattu hakuja tehdessä, mutta jos vastaan tuli samaa asiaa sisältäviä lähteitä, uudempaa julkaisua suosittiin. Kun haku tuotti vain noin 100 tulosta tai alle, oli helppoa käydä otsikot tai tiivistelmät läpi kaikista tuloksista

ja poimia jatkotarkasteluun lupaavat julkaisut. Suoraan tämän työn tutkimuskysymyksiin liittyvien lähteiden mukaanottoehdot olit seuraavat:

- Lähteessä kerrotaan suoraan suorituskyvyntestaustyökalujen valintaan liittyvistä tekijöistä tai lähteessä vertaillaan eri suorituskyvyntestaustyökaluja
- Lähde on vertaisarvioitu, tai muuten korkeasti arvostettu
- Lähteen aihe ei ole tämän työn rajausten ulkopuolella

Muuten korkeasti arvostetun lähteen ehdon toteutti esimerkiksi Molyneauxin (2015) kirja, johon oli viitattu erittäin monessa julkaisussa. Tämän työn rajausten sisäpuolelle laskettiin lähteet, joissa tarkastellaan web-sovelluspalveluihin liittyvien sovellusten suorituskyvyntestaamista. Web-sovelluspalvelurajoite karsi pois lähteitä, joissa tarkasteltiin esimerkiksi robottien tai pelkkien algoritmien suorituskyvyntestausta. Rajausten ulkopuolelle jäi myös lähteet, joissa oli keksitty jokin aivan oma muista poikkeava järjestelmä suorituskyvyntestaustyökalujen vertailuun. Kyseisissä lähteissä oli kehitetty esimerkiksi täysin matemaattinen malli suorituskyvyntestaustyökalujen vertailuun. Kutenkin tämän työn aiheen paisumisen estämiseksi täytyi rajata pois lähteet, joissa työkalunvalintaan oli kehitetty jokin muu menetelmä, kuin eri valintaan vaikuttavien tekijöiden pohtiminen.

Taustatietojen tai tekijöiden tärkeyttä perustelevia lähteitä etsittäessä hakutuloksia oli tuhansia. Tästä syystä tuhansien lähteiden läpikäynnin sijaan suosittiin julkaisuja, joihin oli viitattu useasti. Näistä julkaisuista käytiin läpi lähdeluettelot ja pyrittiin pääsemään primäärilähteeseen. Google-haulla etsittyjä lähteitä valittaessa suosittiin monien eri dokumentaatiosivujen sijaan yhteen vetäviä julkaisuja. Yhteen vetäviä Google-haulla löydettyjä lähteitä viitatessa asioiden pätevyys tarkistettiin kuitenkin primäärilähteestä, kuten jonkin standardin omasta dokumentaatiosivusta, ennen kuin lähde valittiin mukaan.

3. WEB-SOVELLUSPALVELUT JA SUORITUSKYKY

Tässä luvussa käydään läpi pohjatiedot työn aiheen ymmärtämiseksi. Ensin perehdytään web-sovelluspalveluihin. Tästä siirrytään suorituskyvyn ymmärtämiseen käsitteenä ja sen testaamiseen. Viimeiseksi kerrotaan taustatiedot itse suorituskyvyntestaustyökaluista.

3.1 Web-sovelluspalvelut

Termi web-sovelluspalvelu, tai varsinkin sen arkikielinen nimitys web-palvelu, sekoitetaan usein muihin samankaltaisiin termeihin, kuten verkkopalveluihin. Termin määritelmä on kuitenkin World Wide Web Consortiumin (W3C) mukaan ohjelmistojärjestelmä, joka mahdollistaa yhteensopivien tietokoneiden vuorovaikutuksen jonkin tietokoneverkon yli. (Haas & Brown, 2004)

Web-sovelluspalveluilla on jokin rajapinta, joka mahdollistaa vuorovaikutuksen muiden järjestelmien kanssa. Vuorovaikutus tapahtuu käyttäen jotain protokollaa. Useimmissa tapauksissa protokollana toimii HTTP (engl. Hypertext Transfer Protocol), mutta protokolla voi olla myös mikä tahansa muu. (Haas & Brown, 2004) Web-sovelluspalvelun tarjoama rajapinta, ja standardisoidun protokollan käyttö, mahdollistaa ohjelmointikielestä ja alustasta riippumattoman kommunikaation järjestelmien välillä.

Muita esimerkkejä usein käytetyistä protokollista web-sovelluspalveluiden kanssa ovat esimerkiksi SOAP (engl. Simple Object Access Protocol), REST (engl. Representational state transfer), MQTT (engl. Message Queuing Telemetry Transport) ja CoAP (engl. Constrained Application Protocol). REST ei ole käytännössä protokolla, vaan arkkitehtoninen tyyli toteuttaa sovellus. REST kuitenkin rinnastetaan usein SOAP:iin, koska ne ovat molemmat hyvin yleisiä tapoja toteuttaa web-sovelluspalvelu. Tästä syystä REST esitellään tässä alaluvussa. MQTT ja CoAP ovat yleisiä IoT-protokollia (engl. Internet of Things) ja ne esitellään tässä alaluvussa yhtenä esimerkkinä monista mahdollisista erilaisista protokollista.

SOAP on vanhempi protokolla, joka luotiin ennen REST:iä. SOAP on REST:iä hitaampi ja raskaampi, koska jokainen SOAP-viesti sisältää siirrettävän datan lisäksi paljon oheistietoa XML-formaatissa (engl. Extensible Markup Language format). Tämä oheistieto muun muassa kuvailee sovellusta, jossa protokolla on käytössä. REST:iä noudattavat viestit sen

sijaan ovat hyvin minimaalisia, ja eivät sisällä mitään oheistietoa siirettävän datan lisäksi. Tämä voi olla sekä hyöty, että haitta, riippuen halutusta käyttötarkoituksesta. (Walker, 2023) MQTT-protokolla toimii TCP:n (engl. Transmission Control Protocol) päällä, kun taas CoAP toimii UDP:n (engl. User Datagram Protocol) päällä (Craggs, 2022). Nämä kaksi tunnettua IoT-protokollaa toimivat oivana esimerkkinä siitä, miten protokollat voivat hyödyntää toisiaan ja toimia toistensa päällä. Aivan kuten SOAP voi toimia esimerkiksi HTTP:n tai jopa SMTP:n (engl. Simple Mail Transfer Protocol) päällä. SMTP on yleisin protokolla, joka mahdollistaa sähköpostien lähettämisen ja vastaanottamisen.

3.2 Suorituskyvyn mittaaminen

Suorituskyky kertoo ohjelman nopeudesta ajan suhteen ja siitä, miten tehokkaasti ohjelma käyttää resursseja. Suorituskykyä ei ole määritelty jollain tietyllä yksiselitteisellä ominaisuuksien joukolla, mutta sitä voidaan kuvata tietyillä mittareilla. Yleisiä mittareita ovat reagointikyky, vasteaika, suoritusteho, saatavuus, skaalautuvuus ja käyttöaste. (Cortellessa ym., 2011, s. 4; Meier ym., 2007, s. 2; Molyneaux, 2015, s. 2–3) Reagointikyky kertoo, miten nopeasti ohjelma pystyy reagoimaan tiettyyn asiaan. Vasteaika kertoo, kuinka kauan tietyllä tehtävällä kestää läpäistä ohjelma. Suoritusteho kertoo, miten paljon tiettyjä tehtäviä ohjelma pystyy suorittamaan ajan hetkessä. Saatavuus kertoo, kuinka suuren osuuden ajasta ohjelma on valmiina ottamaan uuden tehtävän eikä ole jumissa muun tehtävän kanssa. Skaalautuvuus kertoo ohjelman toimintakyvystä, kun käsiteltävän tehtävien tai datan määrää kasvatetaan. Käyttöaste kertoo, kuinka suuren osan ajasta tietty ohjelman osa tekee töitä verrattuna muihin ohjelman osiin.

Edellisessä kappaleessa esitettiin vain teknillinen näkökulma suorituskyvyn määritelmälle. Suorituskyvyn määritelmä riippuu näkökulmasta. Tavalliselle loppukäyttäjälle hyvä suorituskyky tarkoittaa sitä, että ohjelma suorittaa annetut tehtävät ilman havaittavaa hidastumista tai muuta ärsytystä (Molyneaux, 2015, s. 1–2). Tämän kautta päästään kiinni suorituskyvyn testaamisen vaikeuteen. Miten voidaan testata teknillisesti vaikeaa asiaa, jonka onnistuminen on loppujen lopuksi kiinni vain ihmisen havaitsemasta lopputuloksesta?

3.3 Suorituskyvyntestaus

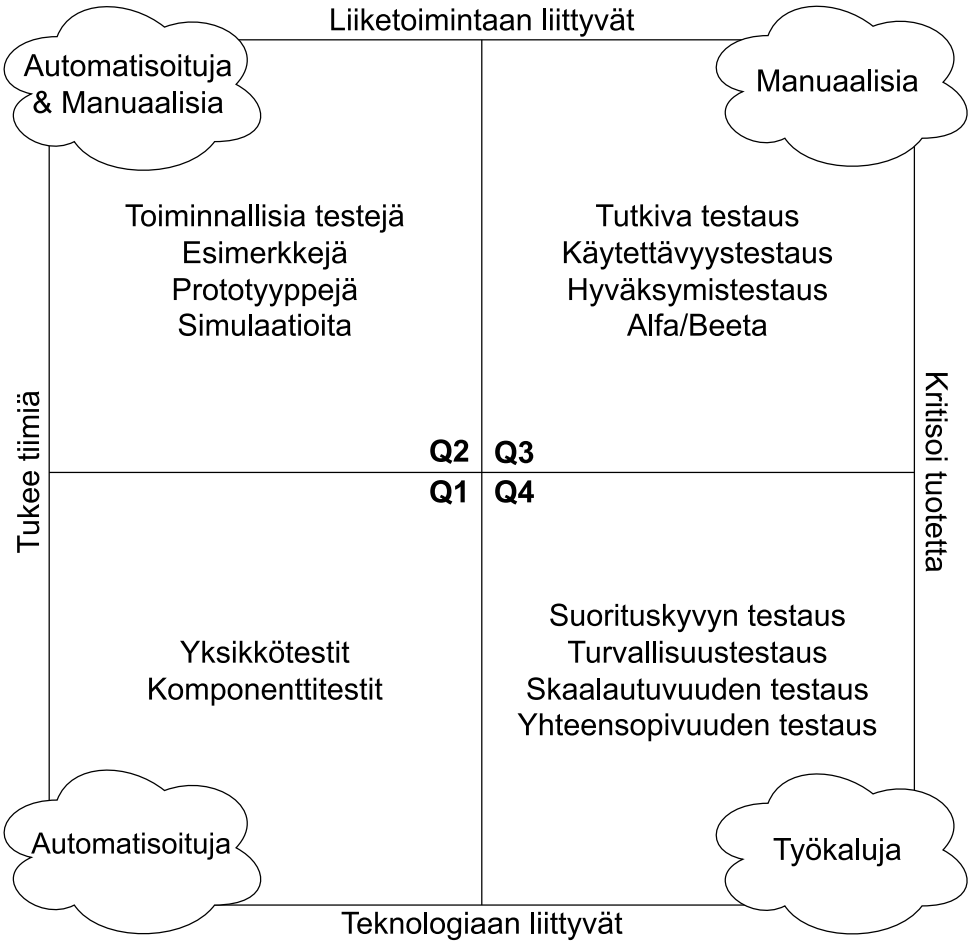
Ohjelmistotestaus on tärkeä osa ohjelmiston kehityksen elinkaarta ja suorituskyvyntestaus on tärkeä osa ohjelmistotestausta (Jacob & Karthikevan, 2018). Suorituskykyä testaamalla voidaan muun muassa arvioida tuotteen valmiutta, arvostella suorituskykyä haluttujen kriteerien perusteella, etsiä suorituskykyongelmien lähteitä tai arvioida tuotteen suorituskykyä eri järjestelmissä tai konfiguraatioissa (Meier ym., 2007). Hyvä suorituskyky on myös tärkeää rahallisesti. Suuryritys Amazon löysi tutkimuksessa, että jokainen sadan millisekunnin viive tuotti yhden prosentin häviön vuotuisissa tuotoissa. Häviö tuotoissa

perustui siihen, että käyttäjät eivät jaksaneet odottaa sivun latautumista, vaan sulkivat sen ja koettivat jotain muuta sivua hitaasti latautuvan sivun sijaan. (Linden, 2006)

Toisin kuin monissa muissa testausmenetelmissä, suorituskyvyn testauksessa testataan ohjelman ei-toiminnallisia vaatimuksia. Luvussa 3.2 mainittuja mittareita voidaan pitää ei-toiminnallisina vaatimuksia. Esimerkki toiminnallisen ja ei-toiminnallisen vaatimuksen erosta voisi olla viestintäpalvelu, jossa lähetetään viesti. Toiminnallinen vaatimus voisi olla, että viestin pystyy lähettämään ja vastaanottaja saa sen viestin. Ei-toiminnallinen vaatimus voisi olla, että järjestelmä pystyy skaalautumaan 10 000 yhtäaikaisen käyttäjän viestittelyyn.

Suorituskyvyntestaus on yleisnimitys monelle eri suorituskyvyntestaustyypille. Eri suorituskyvyntestaustyyppejä ovat muun muassa rasitustestaus (engl. stress testing), kuorman testaus (engl. load testing), vakaustestaus (engl. soak testing tai stability testing), skaalautuvuuden testaus (engl. scalability testing) ja piikkitestaus (engl. spike testing). Rasitustestauksessa rasitetaan ohjelmaa niin paljon, kunnes jokin menee rikki. Näin saadaan selville rajat, joissa ohjelma voi toimia. Kuorman testauksessa testataan, miten ohjelma toimii erikokoisten kuormien alaisena. Vakaustestauksessa testataan ohjelman vakautta ja toimintakykyä pitkäaikaisen suuren kuorman alaisena. Skaalautuvuuden testauksessa testataan ohjelman kykyä skaalautua kasvavaan käyttömäärään. Piikkitestauksessa testataan ohjelman kykyä suoriutua yllättävästä hetkellisestä suuresta kuormituksesta. (Melkozerova & Rassomakhin, 2020; Shalini, 2017)

Kuvasta 3.1 löytyvä Crispinin ja Gregoryn yleistämä ketterän testauksen neljän kvadrantin kuva vetää yhteen tässä alaluvussa käsitellyt asiat ja antaa oivan siirtymän suorituskyvyntestaustyökalujen käsittelyyn. Nelikentän oikea puoli, joka sisältää tuotetta kritisoivia testejä, liittyy vahvasti tämän työn aiheeseen. Oikealta puolelta löytyy monia tekijöitä, joita huomioidaan luvussa 4, kun pohditaan suorituskyvyntestaustyökalun valintaan liittyviä tekijöitä. Myös itse suorituskyvyntestaus löytyy nelikentän neljännestä kvadrantista, jonka kerrotaan sisältävän tuotetta kritisoivia ja teknologiaan keskittyviä testejä, joihin käytetään työkaluja. Nelikentän vasemmalla puolella olevat testit eivät suoraan liity tämän työn aiheeseen, koska ne ovat ohjelmistotestausprosessissa paljon aiempana kuin suorituskyvyntestaus.



Kuva 3.1. *Ketterän kehityksen testausprosessin neljä kvadranttia. Kuva mukailtu lähteestä Crispin ja Gregory, 2009, s. 97.*

3.4 Suorituskyvyntestaustyökalut

Suorituskyvyntestaustyökalut ovat tietokonesovelluksia, joiden avulla voidaan suorittaa automatisoituja suorituskyvyntestejä. Suorituskyvyntestaustyökaluja on todella monia, ja niitä on ollut olemassa jo yli 30 vuotta. Eri työkaluja voidaan käyttää moniin erityyppisiin suorituskyvyntesteihin. Varsinkin web-sovelluksien testaamiseen työkaluja on monia. Kun siirrytään web-sovelluksien testauksesta muunlaisiin sovelluksiin, kuten esimerkiksi sulautettuihin järjestelmiin tai peleihin, sopivien suorituskyvyntestaustyökalujen määrä vähenee huomattavasti. (Molyneaux, 2015, s. 11–12)

Suorituskyvyntestaustyökalujen tärkeimmät ominaisuudet koostuvat usein neljästä osasta, testien luonti, testien hallinta, testikuorman luonti ja tulosten analysointi (Molyneaux, 2015, s. 12). Eri työkaluissa nämä eri ydintoiminnot on toteutettu monilla eri tavoilla ja eri työkalut pystyvät erilaisiin toimintoihin. Testien luonti voi olla esimerkiksi koodin kirjoittamista, käyttöliittymässä kenttien täyttämistä tai toimintojen nauhoittamista. Testien hallinta sisältää testitapausten eri parametrien ja asetuksien säätämisen, sekä itse testien ajamisen. Testikuorman luontiin vaikuttaa työkalun kyky luoda erilaisia testikuormia. Esi-

merkiksi halutaan luoda kuorma, joka simuloi piikkitestausta, jossa 500 käyttäjää yrittää suorittaa jonkin toiminnon yhtä aikaa. Tulosten analysointiin liittyy toiminnot, joilla työkalu pystyy esittämään keräämänsä tulokset. Tähän voi liittyä erilaisia visualisointeja erilaisten diagrammien muodossa, joista käyttäjä voi tehdä itse johtopäätöksiä. (Kołtun & Pańczyk, 2020) Näiden neljän ydintoiminnon lisäksi työkalut voivat sisältää myös muita moduuleja. Nämä perustoimintojen ulkopuolella olevat lisämoduulit usein täydentävät perustoimintojen ominaisuuksia. (Molyneaux, 2015, s. 13) Lisämoduuli voisi olla esimerkiksi ohjelmointirajapinta (engl. Application Programming Interface, API), jonka avulla työkalu voidaan liittää muihin järjestelmiin.

Perimmäinen syy työkalujen käyttöön testauksessa on testien automatisointi. Testien automatisoinnilla on monia perinteisiä hyvin tunnettuja etuja manuaaliseen testaukseen nähden, kuten esimerkiksi testauksen nopeuttaminen tai rahan säästäminen (Crispin & Gregory, 2009, s. 528). Automatisoinnin tuomien perinteisten hyötyjen lisäksi suorituskyvyntestauksessa on erityisiä hyötyjä työkalujen käytöstä. Useimmat luvussa 3.2 mainituista mittareista ovat hyvin vaikeita, tai jopa mahdottomia, mitata manuaalisesti. Esimerkiksi manuaalinen skaalautuvuuden testaaminen olisi hyvin vaikeaa, kun haluttaisiin kasvattaa yhtäaikaista käyttäjien määrää suureksi. Suorituskyvyntestaustyökalu sen sijaan voi simuloida hyvinkin suuria määriä yhtäaikaisia käyttäjiä ongelmitta.

4. SUORITUSKYVYNTESTAUSTYÖKALUN VALINTA

Suorituskyvyntestaustyökalun valinta voi olla vaikeaa etenkin vaihtoehtojen paljouden takia. Valintaan vaikuttavia tekijöitä on paljon ja eri työkaluja voi olla vaikea vertailla keskenään. Tässä luvussa käydään läpi kirjallisuuskatsauksen tuottamat löydökset ja esitellään tärkeimmät työkalun valintaan vaikuttavat tekijät. Tekijät on jaettu selkeyden vuoksi kahteen eri luokkaan, organisatorisiin ja työkalukohtaisiin tekijöihin.

4.1 Organisatoriset tekijät

Tässä alaluvussa käydään läpi suorituskyvyntestaustyökalun valintaan vaikuttavia tekijöitä, jotka eivät suoraan liity tiettyyn työkaluun. Näitä tekijöitä mietitään erityisesti yrityksen tai organisaation näkökulmasta, ja niitä voi pohtia yleispätevästi ennen kuin miettii tiettyjä työkaluja.

4.1.1 Hinta ja kustannukset

Suorituskyvyntestaustyökalut voi jakaa hinnoittelun perusteella kahteen luokkaan, maksullisiin ja maksuttomiin. Maksuttomat työkalut ovat usein myös avoimen lähdekoodin työkaluja, mikä voi tuoda lisämahdollisuuksia mukauttamiseen. Toisaalta suorituskyvyntestaustyökalujen vertailuissa on usein päädytty tuloksiin, joissa monet vertailtavat osa-alueet ovat paremmin toteutettuja maksullisissa työkaluissa (Kołtun & Pańczyk, 2020; Shalini, 2017).

Jos valitaan maksullinen työkalu, täytyy miettiä tarkkaan, miten paljon on valmis maksamaan. Työkaluilla on monia erilaisia lisensointimalleja. Työkalun alkumaksun lisäksi täytyy ottaa huomioon sen käyttö- ja ylläpitokulut. Työkalujen hinnoittelu riippuu usein käyttäjän tarpeista. Useimmiten työkaluja tarjotaan ohjelmistopalveluina (engl. Software as a Service, SaaS), jolloin hinta riippuu pääosin käyttötunneista ja testikuormien suuruudesta (”LoadNinja Pricing”, 2023; ”WebLOAD Pricing”, 2023). Ohjelmistopalveluna ostetulla työkalulla alkumaksu on pieni, mutta työkalun käyttö tuottaa jatkuvia kuluja. Toisaalta työkaluja myydään myös käyttöönotettavaksi paikan päällä, jolloin päästään eroon käyttötuntien hinnoittelusta. Tällöin kuitenkin alkumaksu on suuri ja täytyy myös itse huolehtia työkalun, ja sen vaatimien palvelimien, ylläpitokuluista. (”Gatling Enterprise Pricing”, 2023)

Maksullisissa suorituskyvyntestaustyökaluissa ohjelmistopalvelun tai paikan päällä käytettävän työkalun valinta riippuu siis enimmäkseen käytön paljoudesta. Tämän takia perussääntönä toimii, että ohjelmistopalvelu on usein halvempi, jos suorituskyvyntestejä ei ajeta kovin usein (Molyneaux, 2015, s. 16).

4.1.2 Tarpeen suuruus

Kuten edellisessä alaluvussa todettiin, suorituskyvyntestaustyökalun hinta riippuu eniten tarpeen suuruudesta. Tästä syystä on hyvä miettiä miten paljon tiettyä suorituskyvyntestaustyökalua tulee tarvitsemaan, ennen kuin sitoutuu sen käyttöön. Työkalun hankkiminen voi olla pitkäaikainen investointi ja toiseen työkaluun vaihtaminen tulevaisuudessa voi tuottaa ison työn. Täytyy miettiä tarkkaan mitkä kaikki projektit tarvitsevat suorituskyvyntestausta ja voisiko yksi työkalu kattaa ne kaikki. Tulevaisuuden suunnitelmat tulee myös miettiä tarkkaan. Saatetaanko projekteja laajentaa tulevaisuudessa? Voivatko testitarpeet muuttua? Näitä kysymyksiä on hyvä miettiä luvussa 4.2 mainittujen työkalukohtaisten tekijöiden kautta. Esimerkiksi tulevaisuudessa voi nousta tarve tietyn protokollan käyttöön, mutta tietty työkalu ei välttämättä tue tätä protokollaa.

Pitkäaikaiseen tarpeeseen soveltuvaa suorituskyvyntestaustyökalua valittaessa on tärkeää tutustua työkalun kypsyyteen. Monet maksulliset työkalut ovat kovin uusia ja kärsivät kypsyyden puutteesta (Kołtun & Pańczyk, 2020). Kypsyyttä voi analysoida muun muassa työkalun iän, päivitysten ja tuottojen kautta. Jos työkalu on jo monta vuotta vanha, sillä on todennäköisesti vahva perusta. Kuitenkin täytyy myös tutkia saako työkalu enää päivityksiä. Kehitetäänkö työkalua enää aktiivisesti? Onko myyjä aktiivisesti yhteydessä käyttäjiin esimerkiksi uutisten tai tulevaisuuden suunnitelmien muodossa? Maksullisen työkalun tapauksessa sen takana olevan yrityksen tuotot voivat myös kertoa tarinan, jota ei näe ulospäin. Jos yrityksen tuotot ovat kovassa laskussa, tämä voi kertoa työkalun olevan elinkaarensa lopussa.

4.1.3 Saatavilla oleva tuki

Suorituskyvyntestaustyökalun käyttäminen ei välttämättä ole helppoa. Eri työkalujen käytettävyydessä on suuria eroja, kuten luvussa 4.2.3 on kerrottu. Mahdollisuus teknisen tuen saamiseen voi olla todella tärkeää, riippuen käyttäjän taidoista ja tehtävien testien monimutkaisuudesta. Ilmaisissa avoimen lähdekoodin työkaluissa tekninen tuki on usein saatavilla yhteisön muodossa. Maksullisissa työkaluissa sen sijaan tekninen tuki on usein saatavilla asiakaspalvelijan muodossa lisämaksua vastaan. Taulukossa 4.1 on vertailtu Stack Overflow -sivustosta löytyvien kysymysten lukumäärää, kun hakusanana käytetään eri suorituskyvyntestaustyökalujen nimiä. Tulokset on haettu 25. maaliskuuta, 2023, osoitteesta ”https://stackoverflow.com/search“. Stack Overflow on maailman suosituin ohjelmointiaiheinen kysymys–vastaus-sivusto.

Taulukko 4.1. *Eri työkaluja koskevien kysymysten lukumäärä Stack Overflow -sivustossa*

Hakusana	Kysymysten määrä	Hinnoittelu
JMeter	47 710	Maksuton
Gatling	4 193	Maksuton
LoadRunner	3 305	Maksullinen
WebLOAD	93	Maksullinen
SmartMeter	82	Maksullinen
Kobiton	25	Maksullinen
Silk Performer	20	Maksullinen
HeadSpin	15	Maksullinen
LoadNinja	3	Maksullinen

Taulukossa JMeter ja Gatling ovat maksuttomia avoimen lähdekoodin suorituskyvyntestaustyökaluja. Kaikki loput työkalut ovat maksullisia. Taulukosta nähdään selkeästi, miten yhteisön tuki on valtavasti suurempi avoimen lähdekoodin työkaluilla. Maksullisista työkaluista ainoastaan erittäin suositulla LoadRunner-työkalulla, joka on julkaistu 30 vuotta sitten, on yli sata kysymystä.

Jos työkalun käyttöön ei ole vahvaa yhteisön tukea, täytyy varmistaa, että myyjän tarjoamat oppimisresurssit ovat riittäviä. Kuinka hyvä ja kattava dokumentaatio työkalulla on? Tarjoaako myyjä koulutusta tai muita opetusmateriaaleja? Opetusmateriaaleja työkalun käyttöön voi myös löytyä myyjän ulkopuolelta, kuten kolmannen osapuolen verkkokursseista. On myös tärkeää pitää työntekijöiden osaaminen mielessä (Molyneaux, 2015, s. 15). Tarvittava osaaminen tiettyyn työkaluun saattaa jo löytyä organisaatiosta. Vaihtoehtoisesti jo valmiiksi osaavan työntekijän voi palkata, jos uuden työntekijän palkkaus on aiheellista.

4.2 Työkalukohtaiset tekijät

Suoraan suorituskyvyntestaustyökaluihin liittyviä tekijöitä on paljon enemmän kuin organisatorisia tekijöitä. Tässä alaluvussa käydään läpi web-sovelluspalveluiden kannalta tärkeimmät tekijät liittyen suorituskyvyntestaustyökalun valintaan. Näitä tekijöitä mietittäessä ei enää pohdita yleisellä tasolla, vaan tutustutaan tiettyyn työkaluun ja sen tarjoamiin ominaisuuksiin.

4.2.1 Tarvittava testaustyyppi

Suorituskyvyntestaustyyppejä on monia erilaisia. Luvussa 3.3 esitettiin viisi eri testaustyyppiä ja joissain lähteissä testaustyyppejä on esitelty vieläkin enemmän. Työkalua valittaessa täytyy miettiä, minkä tyyppisiä suorituskyvyntestejä halutaan ajaa. Tämä sitoutuu lukuun 4.1.2, jossa pohdittiin tarpeen suuruutta. Kaikki työkalut eivät sovellu yhtä hyvin kaikkiin testityyppeihin (Memon ym., 2018). Esimerkiksi jos mitataan vasteaikaa verkkosivun lataamisessa, eri työkaluilla saadut tulokset voivat poiketa toisistaan huomattavasti. Yksi työkalu voi laskea mukaan kuvien ja muiden verkkosivun resurssien lataamiseen menevän ajan, kun taas toinen ei laske. (Srivastava, 2021) Myös työkalujen arkkitehtuuri, tai niiden monien erilaisten asetuksien säädöt, voivat aiheuttaa eroja mitatuissa suorituskyvyissä. Esimerkiksi jokin työkalu voi olla kehitetty ohjelmointikielellä, joka vaatii koodin ajamisen virtuaalikoneessa. Java-ohjelmointikieli on tästä yleinen esimerkki, koska Java-ohjelmat ajetaan aina Java-virtuaalikoneessa. Java-virtuaalikone, yleisemmin tunnettu lyhenteellä JVM (engl. Java virtual machine), hyväksyy monia parametreja, jolla voi säätää virtuaalikoneen toimintaa. Nämä voivat vaikuttaa testituloksiin suuresti. (Suffian & Fahrurazi, 2012)

Työkalujen eroavaisuudet tietyissä testitapauksissa eivät kuitenkaan välttämättä ole este tietyn työkalun käyttöön. Jos ympäristö, jossa testit ajetaan, pidetään samanlaisena, mitatut suorituskyvyt pysyvät vertailukelpoisina toisiinsa nähden. Tällöin voidaan mitata suhteellisia muutoksia eri testiajojen välillä, mikä saattaa olla juuri sitä, mitä halutaan tehdä. Tällaisella testauksella voidaan tarkkailla esimerkiksi koodimuutosten vaikutusta ohjelman suorituskykyyn. Kuitenkin kaikissa testaustyypeissä ei pysty turvautumaan suhteelliseen vertailuun. Tästä hyvänä esimerkkinä toimii skaalautuvuuden testaus. Ohjelmistopalveluina tarjottavat pilvessä ajettavat suorituskyvyntestauspalvelut pystyvät kaikista parhaiten skaalautuvuuden testaukseen (Sarojadevi, 2011). Pilviarkkitehtuurin ansiosta testikuorman kokoa voidaan kasvattaa lähes rajatta. Paikan päällä käyttöönotettavissa työkaluissa sen sijaan tulee vastaan omien palvelimien rajat tai itse työkalun rajat.

4.2.2 Yhteensopivuus ja mukautettavuus

Tarvittavien testaustyyppien jälkeen päästään luonnollisesti pohtimaan suorituskyvyntestaustyökalun yhteensopivuutta ylipäätänsä. Tähän liittyy tarvittavien protokollien tukeminen, teknologioiden yhteensopivuus ja työkalun toimintojen laajentamisen mahdollisuus. Web-sovelluspalveluja testatessa yleisimmät protokollat ovat HTTP(S) ja SOAP. Nämä protokollat ovat laajasti tuettuja, joten tilanne on siltä kannalta hyvä. Kuitenkin protokollia on monia muita ja työkalut tukevat hyvin vaihtelevia määriä niistä. (Pathak, 2023) Tästä syystä protokollien tuki kannattaa aina varmistaa.

Teknologioiden yhteensopivuuteen kannattaa kiinnittää huomiota, jos suorituskyvyntestaukseen liittyy tiettyjä sovelluskehyksiä (engl. framework) tai tiettyjen teknologioiden joitain

erityispiirteitä. Hyvänä esimerkkinä näistä toimii taas Java-ohjelmointikieli. Java-pohjaisten sovellusten testaamisessa on erityisen tärkeää päästä kiinni sovelluksen sisäisiin toimintoihin (Molyneaux, 2015, s. 208). Java-sovelluksen sisäisistä toiminnoista voidaan ottaa esimerkiksi Java-oliot (engl. java object). Java-pohjaiset työkalut voivat tukea Java-olioita, mistä voi olla erityistä hyötyä testauksessa. Tällöin työkalu pystyy monitoroimaan tarkasti Java-olioita, mikä voi mahdollistaa esimerkiksi muistivuotojen löytämisen.

Viimeisin yhteensopivuuteen liittyvä tekijä, jota on syytä pohtia, on luvussa 3.4 mainittu lisämoduulien tuki. Lisämoduuleilla mahdollistetaan suorituskyvyntestaustyökalun toiminnallisuuksien laajentaminen. Aiemmin mainittu ohjelmointirajapinta on erittäin vahva esimerkki hyvästä lisämoduulista, joka nostaa työkalun arvoa. Ohjelmointirajapinnan avulla työkalu voidaan yhdistää omiin sovelluksiin, mikä mahdollistaa esimerkiksi testidatan reaaliaikaisen lukemisen työkalusta. (Molyneaux, 2015, s. 143) Yhteensopivuuteen keskittyviä lisämoduuleja voi olla myös valmiiksi rakennettu sisään sovellukseen. Laajasti käytettyjen kolmannen osapuolen sovelluksia yhdistävät lisämoduulit ovat yleinen esimerkki sisäänrakennetusta lisämoduulista. (Tricentis, 2022) Näistä sovelluksista tärkeimpiä esimerkkejä ovat DevOps-työkalut (engl. Software Development and IT Operations tools).

Jatkuvan integraation hyödyntäminen ohjelmiston testausprosessissa on välttämätöntä, jos haluaa menestyä ohjelmistokehitysyrityksenä (Crispin & Gregory, 2009, s. 486). Automatisoitujen suorituskyvyntestien lisääminen jatkuvaan integraatioon ja julkaisuun (engl. Continuous Integration and Continuous Delivery, CI/CD) voi olla itsestäänselvyydeltä kuulostava asia, mutta työkalut pystyvät tähän vaihtelevalla tasolla. Jotkin työkalut pystyy yhdistämään suoraan kaikkiin yleisiin DevOps-työkaluihin, kun taas toisissa ei ole sisäänrakennettua tukea ja yhdistäminen voi olla huomattavasti vaikeampaa. (Pathak, 2023) Tästä syytä omien DevOps-työkalujen ja -harjoitteiden yhteensopivuuden varmistaminen on tärkeää.

4.2.3 Käytettävyys ja toiminnot

Käytettävyys viittaa sovelluksen helppokäyttöisyyteen ja loppukäyttäjän tyytyväisyyteen sovelluksen käyttökokemuksesta (Hilbert & Redmiles, 2000). Koska suorituskyvyntestaustyökalut ovat sovelluksia, niitä vertailtaessa käytettävyys on tärkeää ottaa huomioon. Tämä oli huomioitu useassa työkaluja vertailevissa lähteissä (Kołtun & Pańczyk, 2020; Melkozerova & Rassomakhin, 2020; Memon ym., 2018). Suorituskyvyntestaustyökalun käytettävyyttä arvioitaessa voidaan ottaa huomioon mitkä tahansa työkalun ominaisuuksista. Aiheen liian suureksi paisumisen välttämiseksi otetaan tarkasteluun vain luvussa 3.4 löydetyistä työkalun ydintoiminnoista kaksi tärkeintä, testien luonti ja tulosten analysointi. Muita tarkasteltavia ominaisuuksia voisivat olla esimerkiksi testien hallinta, testikuorman luonti tai työkalun yhteensopivuus jonkin ulkoisen kolmannen osapuolen sovelluksen kanssa.

Eri suorituskyvyntestaustyökalut tarjoavat monia erilaisia tapoja luoda testejä (Molyneaux, 2015, s. 15). Käytettävyys näiden eri tapojen, ja niiden toteutusten, välillä voi vaihdella suuresti (Scarpino, 2012). Käytettävyyserot testien luomisen välillä voivat johtua esimerkiksi vaikeakäyttöisyydestä, toiminnallisuuksien puutteesta tai koodikielestä. Kołtunin ja Pańczykin (2020) tutkielmassa vertailtiin kolmea eri suorituskyvyntestaustyökalua. He löysivät vaikeakäyttöisyyttä erään työkalun testienluontimoduulin konfiguroinnista. Toinen vertailtavissa ollut työkalu toimi suoraan ilman mitään konfigurointia. Toiminnallisuuksien puutteeseen liittyen tutkielmassa painotettiin mahdollisuutta nauhoittaa testejä. Web-sovelluspalveluita testatessa asiat liittyvät usein suoraan loppukäyttäjän toimenpiteisiin. Nauhoittamalla voi helposti luoda testin, joka esimerkiksi kuvastaa yleisiä loppukäyttäjän toimenpiteitä sovelluksessa. Toinen ominaisuuden puute, jota korostettiin tutkielmassa, oli parametrisoinnin puute tai huonous. Parametrisointi kuvastaa työkalun kykyä ottaa vastaan syötettä, jota käytetään testien ajamisessa. Parametrisointiin voidaan käyttää esimerkiksi suurta tietokantaa, joka sisältää esimerkiksi tiettyjä käyttäjätietoja, joita halutaan käyttää testiajossa. Yhdellä vertailussa olleella työkalulla oli todella huono tuki parametrisointiin. Koodikieliin liittyen kaksi vertailtavista työkaluista käytti täysin eri koodikieliä testien luontiin. Kolmas työkalu ei käyttänyt koodikieltä ollenkaan, vaan kaikki tapahtui graafisen käyttöliittymän kautta.

Testien luomisen ja niiden ajon jälkeen päästään suorituskyvyntestaustyökalujen ehkä tärkeimpään toiminnallisuuteen, tulosten analysointiin. Sovellusten yksikkötestaamisesta tuttua läpäisty–hylätty-asteikkoa harvoin käytetään suorituskyvyntestien kanssa. Suorituskyvyntesteistä saadaan usein ulos raakaa dataa, jota pitää analysoida ja käsitellä jollain tavalla. Visualisoinnit joidenkin diagrammien muodossa ovat erinomaisia tähän, koska ne parantavat sovelluksen käytettävyyttä huomattavasti (Hilbert & Redmiles, 2000). Esimerkiksi ajan yli esittämiseen kuvaajat ovat havainnollistavia, tai eri komponenttien suhteellista resurssien kulutusta on kätevä havainnollistaa ympyrädiagrammilla. Erilaiset saatavilla olevat visualisoinnit ja muut datan esitysmuodot vaihtelevat suuresti eri työkalujen välillä (Srivastava, 2021). Tästä syystä datan esitysmuotoihin kannattaa kiinnittää huomiota suorituskyvyntestaustyökalua valittaessa. Hyvien visualisointien puute ei kuitenkaan välttämättä ole este työkalun käyttöön. Jos työkalussa on hyvät mahdollisuudet datan vientiin, esimerkiksi ohjelmointirajapinnan kautta, voi työkalun yhdistää johonkin ulkopuoliseen ohjelmaan, joka hoitaa datan visualisoinnin.

5. TULOSTEN TARKASTELU

Tässä luvussa tarkastellaan löydettyjä tuloksia. Tuloksien pätevyyttä ja luotettavuutta pohditaan vertailemalla niitä aiempiin tuloksiin ja esittämällä kritiikkiä niitä kohtaan. Työn tulokset ovat erilaisten suorituskyvyntestaustyökalujen valintaan vaikuttavien tekijöiden muodossa. Tekijät, jotka löydettiin kirjallisuuskatsauksen avulla, löytyvät tämän työn tulosluvusta 4. Luvussa esitetyt tekijät jaettiin selkeyden vuoksi kahteen eri luokkaan. Ensimmäinen luokka oli organisatoriset tekijät ja toinen luokka oli työkalukohtaiset tekijät.

Organisatoristen tekijöiden alaluvussa löydetyt tekijät jaettiin kolmeen kategoriaan, hintaan, työkalun tarpeen suuruuteen ja saatavilla olevaan tukeen. Hintatekijöissä tietoa perusteltiin eri työkalujen hinnoittelusivuilta löytyviin tietoihin ja numeroihin. Kuitenkin monen työkalun hinnoittelusivussa luki myös mahdollisuus ottaa myyjään suoraan yhteys ja tehdä oma sopimus. Tätä ei otettu huomioon, eikä työssä ole tietoa mahdollisten omien sopimusten sisällöstä. Nämä tuntemattomat tekijät voisivat vaikuttaa joihinkin johtopäätöksiin, jotka tehtiin alaluvussa 4.1.1. Työkalun käytön tarvetta pohdittaessa käytettiin paljon omaa pohdintaa, jota ei ollut perusteltu lähteisiin. Tieto kuitenkin on luonnostaan yleispätevää ja sitä sivuttiin monissa eri lähteissä. Työkaluille saatavilla olevaa tukea käsiteltäessä esitettiin taulukko, jossa vertailtiin Stack Overflow -sivustosta löytyviä kysymysten määrää eri työkalujen välillä. Taulukko antoi selvät tulokset, joista tehtiin johtopäätöksiä. On syytä kuitenkin ottaa huomioon, että Stack Overflow on vain yksi sivusto. On mahdollista, että tietyistä työkaluista puhutaan hyvinkin paljon joissain muissa sivustoissa.

Työkalukohtaisten tekijöiden alaluvussa löydetyt tekijät jaettiin myös kolmeen kategoriaan, testaustyyppeihin, yhteensopivuuteen ja käytettävyyteen. Tarvittavia testaustyyppejä tarkasteltaessa joitain päätelmiä tehtiin perustuen lähteisiin, jotka ovat yli 10 vuotta vanhoja. Tässä työssä on monia lähteitä, jotka eivät välttämättä edusta alan kaikista uusinta tietoa. Koska tämän työn tuloksia perustellaan lähteillä, eikä ajankohtaisilla henkilökohtaisilla alan ammattilaisen kokemuksilla, on syytä miettiä lähteiden pätevyyttä. Kuitenkin vanhemmista lähteistä löydetyt tiedot ovat usein yleispäteviä, joten on hyvin mahdollista, että tieto on yhä aiheellista. Yhteensopivuustekijöitä käsiteltäessä olisi hyvä olla joitain konkreettisia esimerkkejä siitä, miten suorituskyvyntestaustyökalujen toiminnallisuuksia voidaan laajentaa. Esimerkiksi olisi voitu kehittää oma lisämoduuli, jolla voisi demonstroida tietyn puuttuvan ominaisuuden lisäämistä työkaluun. Kuitenkin kokeellisen osuuden tuottaminen menee yli tämän työn tavoitteiden ja rajausten. Käytettävyyttä käsittelevä alaluku antaa

myös esimerkin tässä työssä tehdyistä rajauksista, jotka estävät aiheen kasvamisen liian suureksi. Suorituskyvyntestaustyökalujen käytettävyydestä voisi tarkastella hyvinkin monta eri asiaa. Kuitenkin rajausten takia alaluvussa päädyttiin käsittelemään vain työkalujen kahta ydintoimintoa, testien luontia ja tulosten analysointia.

Työn tuloksia voidaan käsitellä listana tekijöitä, joita pitää huomioida suorituskyvyntestaustyökalua valittaessa. Tällöin tämän työn tuloksia voidaan vertailla luvussa 1 mainittuihin oppaisiin, joissa on muiden löytämiä listoja tekijöistä. Otetaan vertailuun kaksi Google-haulla löydettyä opasta, joita ei käytetty tämän työn tulosten tuottamiseen. Urbanovichin (2021) ja PerfMatrixin (2020) oppaista löydetyt tekijät on koottu taulukkoon 5.1, jossa niitä vertaillaan tämän työn tuloksiin.

Taulukko 5.1. *Nettioppaissa esiteltyjen valintatekijöiden löytyminen tästä työstä*

Nettiopas	**Valintatekijä**	**Tämä työ**
PerfMatrix; Urbanovich	Protokollien tuki	✓
PerfMatrix; Urbanovich	Eri testaustyypit	✓
PerfMatrix	Tulosten raportointi	✓
PerfMatrix; Urbanovich	Lisenssimallit	✓
PerfMatrix; Urbanovich	Tuki	✓
PerfMatrix; Urbanovich	CI/CD	✓
PerfMatrix; Urbanovich	Yhteensopivuus	✓
Urbanovich	Mukautettavuus	✓
PerfMatrix; Urbanovich	Hinta	✓
PerfMatrix	Omien palvelimien hinta	✓
PerfMatrix	Osaaminen työmarkkinoilla	✓
PerfMatrix	Nauhoitusominaisuus	✓
PerfMatrix	Parametrisointi	✓
PerfMatrix	Käytettävyys	✓
PerfMatrix	Tapahtuma & pyyntö	✗
PerfMatrix	Vastauksen vahvistaminen	✗
PerfMatrix	Reaaliaikainen monitorointi	✓

Taulukossa näkyvän vertailun perusteella voidaan sanoa, että tämä työ onnistui hyvin kattavasti löytämään ja käsittelemään tekijöitä, joita oli myös muissa suorituskyvyntestaustyökalun valintaa käsittelevissä töissä. Puuttuvat tekijät eivät tulleet vastaan tämän

työn kirjallisuuskatsauksessa ja jäivät sen takia täysin huomiotta. Tekijöiden puuttuminen tästä työstä voi johtua tämän työn rajauksista, kertoa vajaasta kirjallisuuskatsauksesta tai ehkä tekijät eivät vain ole kovin yleisiä ja ovat päässeet ainoastaan PerfMatrixin henkilökohtaiselle listalle.

Ylipäätänsä tässä työssä jouduttiin asettamaan paljon rajoitteita, jotta aihe pysyy työn laajuustavoitteissa. Tiettyjä asioita käsiteltiin vain rajallisesti ja joitain asioita jätettiin pois heti aluksi, kuten tutkimusmenetelmää kuvaavassa luvussa 2 todettiin. Tästä huolimatta taulukossa 5.1 tehdyn vertailun mukaan työn tulokset olivat hyvin kattavat. Näiden kaikkien päätelmien perusteella voidaan sanoa, että työn tulokset onnistuivat vastaamaan tämän työn tutkimuskysymyksiin.

6. YHTEENVETO

Tässä työssä tarkasteltiin suorituskyvyntestaustyökalun valintaa web-sovelluspalveluille. Suorituskyvyntestaustyökaluja on hyvin monia, joten voi olla haastavaa vertailla eri työkaluja ja valita sopivin vaihtoehto. Asiaan perehdyttiin kirjallisuuskatsauksen keinoin ja pyrittiin löytämään vastaukset tutkimuskysymyksiin:

1. Miten valitaan oikea suorituskyvyntestaustyökalu web-sovelluspalveluiden testaamiseen?
2. Mitä tekijöitä pitää ottaa huomioon, kun vertaillaan eri suorituskyvyntestaustyökaluja?

Kirjallisuuskatsauksen perusteella suorituskyvyntestaustyökalujen valintaa pohditaan erilaisten tekijöiden avulla. Tästä syystä tämän työn ydintulokset koostuivat näiden tekijöiden löytämisestä ja analysoimisesta. Löydetyt tekijät jaettiin selkeyden vuoksi kahteen kategoriaan, organisatorisiin tekijöihin ja työkalukohtaisiin tekijöihin. Organisatoristen tekijöiden alaluvussa perehdyttiin tekijöihin, jotka liittyivät suorituskyvyntestaustyökalujen hinnoitteluun, tarpeen suuruuteen ja saatavilla olevaan tukeen. Työkalukohtaisten tekijöiden alaluvussa perehdyttiin tarvittavaan testaustyyppiin, yhteensopivuuteen ja käytettävyyteen. Löydetyt tekijät, ja täten koko työn tulokset, olivat kattavat ja onnistuivat vastaamaan tämän työn tutkimuskysymyksiin hyvin. Työkalun valintaan vaikuttavia tekijöitä on kuitenkin monia, ja tästä syystä niitä kannattaa tarkastella tilanteen mukaan. Oma tarve saattaa vaatia jotain erikoistunutta ominaisuutta, jota ei mainita missään muiden tekemissä oppaissa. Näistä syistä voidaan todeta, että mitään yleispätevää ohjetta suorituskyvyntestaustyökalun valintaan ei voida mielekkäästi muotoilla.

Tässä työssä suorituskyvyntestaustyökalun valintaan liittyvä pohdinta oli rajattu web-sovelluspalveluihin. Rajaus tehtiin, koska suorituskyvyntestaus, ja siihen käytettävät työkalut, on kokonaisuudessaan liian laaja aihe tämän työn käsiteltäväksi. Rajausten puitteissa karsittiin pois myös lähteitä, joissa työkaluja vertailtiin eri tekijöihin perehtymisen sijaan joillain täysin muilla lähestymistavoilla. Rajausten ulkopuolelle jätetyistä aiheista löytyy vielä mielenkiintoisia jatkotutkimusaiheita. Olisi mielenkiintoista esimerkiksi vertailla eräässä tutkimuksessa kehitettyä matemaattista suorituskyvyntestaustyökalun valintamallia perinteiseen tekijöiden pohtimiseen. Olisi myös mahdollista perehtyä täysin oman työkalun kehittämiseen ja vertailla sen toteuttamiskelpoisuutta ja mahdollisia kustannuksia valmiin työkalun käyttämiseen.

LÄHTEET

Cortellessa, V., Di Marco, A., & Inverardi, P. (2011). What Is Software Performance? Teoksessa *Model-Based Software Performance Analysis* (s. 1–7). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-13621-4_1

Craggs, I. (2022). *SOAP vs REST API*. Haettu haettu 10.4.2023, osoitteesta osoitteesta https://www.hivemq.com/blog/mqtt-vs-coap-for-iot/

Crispin, L., & Gregory, J. (2009). *Agile testing: A practical guide for testers and agile teams*. Addison-Wesley.

Gatling Enterprise Pricing. (2023). Haettu haettu 25.3.2023, osoitteesta osoitteesta https://gatling.io/pricing/#self-hosted

Haas, H., & Brown, A. (2004). *Web Services Glossary*. Haettu haettu 5.3.2023, osoitteesta osoitteesta https://www.w3.org/TR/ws-gloss/

Hilbert, D., & Redmiles, D. (2000). Extracting usability information from user interface events. *ACM computing surveys*, *32*(4), 384–421. https://doi.org/10.1145/371578.371593

Jacob, A., & Karthikevan, A. (2018). Scrutiny on Various Approaches of Software Performance Testing Tools. *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 509–515. https://doi.org/10.1109/ICECA.2018.8474876

Kołtun, A., & Pańczyk, B. (2020). Comparative analysis of web application performance testing tools. *Journal of Computer Sciences Institute*, *17*, 351–357. https://doi.org/10.35784/jcsi.2209

Linden, G. (2006). *Slides from my talk at Stanford*. Haettu haettu 5.3.2023, osoitteesta osoitteesta http://glinden.blogspot.com/2006/12/slides-from-my-talk-at-stanford.html

LoadNinja Pricing. (2023). Haettu haettu 25.3.2023, osoitteesta osoitteesta https://loadninja.com/pricing/

Meier, J., Farre, C., Bansode, P., Barber, S., & Rea, D. (2007). *Performance testing guidance for web applications: patterns & practices*. Microsoft press.

Melkozerova, O. M., & Rassomakhin, S. G. (2020). Software performance testing. *Bulletin of V.N. Karazin Kharkiv National University, series «Mathematical modeling. Information technology. Automated control systems»*, (45), 56–66. https://doi.org/10.26565/2304-6201-2020-45-07

Memon, P., Dewani, A., Bhatti, S., & Hafiz, T. (2018). Scrutinzing Automated Load Testing via Blazmeter, Apache JMeter and HP LoadRunner. *International Journal of Advanced Studies in Computers, Science and Engineering*, *7*(3), 35–42.

Molyneaux, I. (2015). *The Art of Application Performance Testing: From Strategy to Tools*. O'Reilly Media, Incorporated.

Pathak, A. (2023). *Top 27 Performance Testing Tools to Use in 2023*. Haettu haettu 25.3.2023, osoitteesta osoitteesta https://kinsta.com/blog/performance-testing-tools/

PerfMatrix. (2020). *How to choose the right Performance Testing tool?* Haettu haettu 10.4.2023, osoitteesta osoitteesta https://www.perfmatrix.com/how-to-choose-the-right-performance-testing-tool/

Sarojadevi, H. (2011). Performance testing: methodologies and tools. *Journal of Information Engineering and Applications*, *1*(5), 5–13.

Sava, J. A. (2022). *IT budgets - Statistics & Facts*. Statista. Haettu haettu 28.3.2023, osoitteesta osoitteesta https://www.statista.com/topics/6198/it-budgets-and-investments

Scarpino, J. (2012). Evaluating and Implementing Load and Performance Testing Tools to Test Adobe Flex and Other Rich Internet Applications: A Case Study. *Issues in information systems*, *13*(1), 1–10. https://doi.org/10.48009/1_iis_2012_1-10

Shalini, J. T. (2017). Web Performance Testing Tools–A Review. *International Journal for Research in Applied Science and Engineering Technology*, *5*(10), 1057–1063. https://doi.org/10.22214/ijraset.2017.10154

Srivastava, N. (2021). Software and Performance Testing Tools. *Journal of Informatics Electrical and Electronics Engineering (JIEEE)*, *2*(1), 1–12. https://doi.org/10.54060/JIEEE/002.01.001

Suffian, M. D. M., & Fahrurazi, F. R. (2012). Performance testing: Analyzing differences of response time between performance testing tools. *2012 International Conference on Computer & Information Science (ICCIS)*, *2*, 919–923. https://doi.org/10.1109/ICCISci.2012.6297157

Tricentis. (2022). *Integrate NeoLoad with third-party tools* [NeoLoad Documentation]. Versio 9.0. Haettu haettu 25.3.2023, osoitteesta osoitteesta https://documentation.tricentis.com/neoload/9.0/en/WebHelp/#11857.htm

Urbanovich, M. (2021). *Performance testing tools: 8 things to consider*. Haettu haettu 10.4.2023, osoitteesta osoitteesta https://techbeacon.com/app-dev-testing/8-things-consider-about-performance-testing-tools

Walker, A. (2023). *SOAP vs REST API: Difference Between Web Services*. Haettu haettu 10.4.2023, osoitteesta osoitteesta https://www.guru99.com/comparison-between-web-services.html

WebLOAD Pricing. (2023). Haettu haettu 25.3.2023, osoitteesta osoitteesta https://www.radview.com/pricing/