Validoinnissa käytettävät ohjelmistokirjastot: REST-rajapintojen ja JSON-datamuodon merkitys valinnassa

Kimmo Lindeman

Haaga-Helia ammattikorkeakoulu

Liiketalouden ammattikorkeakoulututkinto

Opinnäytetyö

2023

Tiivistelmä

Tekijä(t)
Kimmo Lindeman

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Validoinnissa käytettävät ohjelmistokirjastot: REST-rajapintojen ja JSON-datamuodon merkitys valinnassa

Sivu- ja liitesivumäärä
41 + 7

Opinnäytetyö käsittelee validoinnissa käytettävien ohjelmistokirjastojen valintaa ja REST-rajapintojen sekä JSON-datamuodon merkitystä siinä yhteydessä. Aihe on merkityksellinen monestakin syystä. Lähes kaikki ohjelmistot perustuvat ohjelmistokirjastojen hyödyntämiseen. Ohjelmistokirjaston valinta vaikuttaa moneen tärkeään tekijään kuten ohjelmiston toimintavarmuuteen, tietoturvaan, suorituskykyyn ja lisensointiin. REST on taas yksi yleisimmin käytetyistä rajapinta-arkkitehtuureista ja JSON on hyvin suosittu datamuoto juuri REST-rajapinnoilla. Lisäksi lähes kaikki sovellusten välillä liikkuva data tulisi validoida esimerkiksi tietoturvaongelmien välttämiseksi ja datan eheyden takaamiseksi.

Tutkimuksessa selvitettiin, että millaisiin asioihin kiinnitetään huomiota validointikirjaston valinnassa ja eroaako se merkittävästi muiden ohjelmistokirjastojen valinnasta. Lisäksi tutkittiin, miten REST-rajapinta-arkkitehtuuri huomioidaan ja millainen merkitys JSON datamuodolla on validointikirjaston valinnassa. Tutkimus toteutettiin kyselytutkimuksena ohjelmistoalan ammattilaisille. Kyselyyn oli mahdollista vastata noin kuukauden ajan ja vastaajia rekrytoitiin jakamalla linkkiä sosiaalisen median palveluissa.

Tutkimuksen tulosten perusteella voidaan todeta, että yleiset ohjelmistokirjastojen valintaperusteet soveltuvat yhtä hyvin validointikirjaston valintaan kuin muidenkin ohjelmistokirjaston valintaan. Erot näiden kahden välillä ovat vähäisiä ja tuntuvat perustuvan ohjelmistoalan ammattilaisten mieltymyksiin painottaa valinnassa eri asioita. Validointikirjastojen kohdalla on kuitenkin tunnistettavissa asioita, jotka otetaan erityisesti huomioon valinnassa. Näitä asioita ovat yhteensopivuus, helppokäyttöisyys, skaalautuvuus ja tietoturva sekä se, että validointiin voidaan käyttää ohjelmistokehyksen tarjoamia ominaisuuksia validointikirjaston sijasta.

Tuloksista välittyy selkeä kuva, että REST-rajapinta-arkkitehtuuri tulee ottaa huomioon validointikirjasto valittaessa. Siinä tärkeitä näkökohtia ovat tietoturva, skaalautuvuus ja suorituskyky. Lisäksi käyttömukavuus ja validointikirjaston kyky tuottaa tietotyypit ja validointi ovat huomionarvoisia asioita. JSON-datamuodon merkitys validointikirjaston valintaan on hyvin pieni.

Asiasanat
Ohjelmistokirjastot, validointi, REST, JSON, rajapinnat, API

Sisällys

1 Johdanto

Lähes kaikki ohjelmistot perustuvat ohjelmistokirjastojen hyödyntämiseen. Koska kirjastot nopeuttavat ja helpottavat ohjelmiston kehitystyötä, saattaa yksi ohjelmisto käyttää kymmeniä, tai jopa satoja kirjastoja. Ohjelmistokirjaston valinta muodostaakin tärkeän suunnittelupäätöksen, sillä jokainen käytettävä kirjasto vaikuttaa ohjelmiston suorituskykyyn, toimintavarmuuteen, virheettömyyteen sekä ohjelmiston lisensoitavuuteen ja kustannuksiin.

Ohjelmistot hakevat ajantasaista tietoa ohjelmointirajapintojen kautta, mikä mahdollistaa reaaliaikaisten palveluiden tarjoamisen. Rajapintoja käytetään mm. tiedonsiirtoon sovellusten välillä ja käyttäjän syöttämien tietojen välittämiseen. Rajapinnan avulla siirretyn datan validointi on välttämätöntä lähes jokaisessa sovelluksessa tietoturvaongelmien välttämiseksi ja datan eheyden ja palvelun laadun takaamiseksi.

Tässä opinnäytetyössä käsitellään erityisesti REST rajapinta-arkkitehtuuria, jota käyttää jopa 93,5 prosenttia kehittäjistä (Simpson 2022). Työssä selvitetään, minkälaisia perusteita käytännön ohjelmistokehitystyössä toimivat ammattilaiset käyttävät ohjelmistokirjastojen valinnassa. Erityinen painoarvo on rajapinnan kautta hankittavan datan validointiin käytettävillä kirjastoilla ja sillä, kiinnittävätkö sovelluskehittäjät huomiota rajapinnan tarjoaman datamuotoon tehdessään oman ohjelmistonsa arkkitehtuurivalintoja. Tätä selvitetään tekemällä lomakekysely ohjelmistoalalla työskenteleville henkilöille, etsien vastauksia seuraaviin tutkimuskysymyksiin:

Mihin asioihin kiinnitetään huomiota validointiin käytettävän ohjelmistokirjaston valinnassa?
Eroaako validointikirjastojen valinta muiden ohjelmistokirjastojen valinnasta?
Miten REST-rajapinta-arkkitehtuuri tulee huomioida validointikirjaston valinnassa?
Millainen merkitys on JSON-datamuodolla validointikirjaston valinnassa?

Opinnäytetyön tavoitteena on löytää käytännön kehitystyöhön soveltuvat perustelut ohjelmistokirjaston valinnalle. Tieto ohjelmistoalan ammattilaisten valintaperusteista tekee ohjelmistokirjastojen valinnasta helpompaa uraansa aloittavalle ohjelmistokehittäjälle. Tuloksia voidaan soveltaa hyödyntää myös ihmiset, jotka ovat työnsä puolesta tekemisissä ohjelmistoprojektien kanssa, mutta eivät osallistu aktiivisesti kehitystyöhön. Esimerkiksi ohjelmistoja tilaavat henkilöt voivat joutua kiinnittämään huomioita ohjelmistokirjastojen valintaan vaikkapa korkeiden suorituskykyvaatimusten tai lisensoinnin vuoksi. Kuten sanottu ohjelmistokirjastot ja datan validointi ovat tärkeitä lähes jokaisessa ohjelmistoprojektissa, joten ei ole yhdentekevää millä perusteilla ohjelmistokirjastot valitaan.

Sanasto

JSON	Lyhenne tulee englannin kielen sanoista JavaScript Object Notation. JSON on datamuoto, jota käytetään tiedon siirtämiseen järjestelmästä toiseen. JSON:ssa tiedonjärjestely perustuu nimi ja arvo pareihin, jotka jäsennetään puurakenteeseen. JSON perustuu JavaScript ohjelmointikielen standardiin, mutta se on riippumaton ohjelmointikielestä. JSON:ssa tieto on ihmisen luettavassa muodossa.
Ohjelmistokirjasto	Kokoelma uudelleen käytettävää jonkun toisen kirjoittamaa ohjelmakoodia, jonka tarkoituksena auttaa ohjelmistokehittäjää ohjelmiston rakentamisessa ja ohjelmointikielen kääntäjää ohjelmakoodin suorittamisessa. Ohjelmistokirjasto koostuu luokista, funktioista, konfiguraatiotiedoista tai muista tarvittavista osista.
Ohjelmointirajapinta	API (Application Programming Interface) eli ohjelmointirajapinta määrittelee, miten ohjelmisto tarjoaa tietoja tai palveluita muille sovelluksille tai järjestelmille. Ohjelmistorajapinta on joukko komentoja, toimintoja, protokollia ja objekteja, joita voidaan käyttää ohjelmistojen luomiseen tai vuorovaikutukseen ulkoisen järjestelmän kanssa.
REST	REST lyhenne tule sanoista Representational State Transfer. Se on arkkitehtuurinen tyyli ohjelmointirajapinnan toteuttamiseen ja sen avulla tarjotaan pääsy sovelluksen resurssein. REST hyödyntää tilatonta http-protokollaa ja sen metodeja kuten GET, POST, PUT ja DELETE. Data-muotona on usein JSON, mutta REST voi käyttää muita esitysmuotoja kuten HTML tai XML.
Skeema	Skeema on tietomalli, joka määrittelee, että miten data on järjestettyä. Skeeman avulla määritellään sallitut tietotyypit ja arvoalueet sekä niiden väliset yhteydet. Skeeman avulla voidaan luoda tiedoille validointisääntöjä, joiden perusteena ovat edellä mainitut tietotyypit ja arvoalueet.
Validointi	Tietojen validointi on prosessi, jossa verrataan johonkin järjestelmään syötettyjä tietoja ennalta määriteltyihin vaatimuksiin. Validointi sisältää sarjan tarkistuksia, jotka voivat olla yksinkertaisia tarkistuksia tai

	strukturoituja ehdollisia tarkistuksia. Esimerkkinä voisi olla, että tuntipalkan pitää olla positiivinen desimaali- tai kokonaisluku.
XML	Lyhenne muodostuu sanoista Extensible Markup Language. XML on merkintäkielen standardi, jossa tiedot jäsennetään loogisen rakenteen mukaan. Tiedostomuotona sitä käytetään tiedonsiirtoon ja datan tallentamiseen. Tieto jäsennetään puurakenteeseen.

2 Tietoperusta

2.1 Ohjelmointirajapinnat

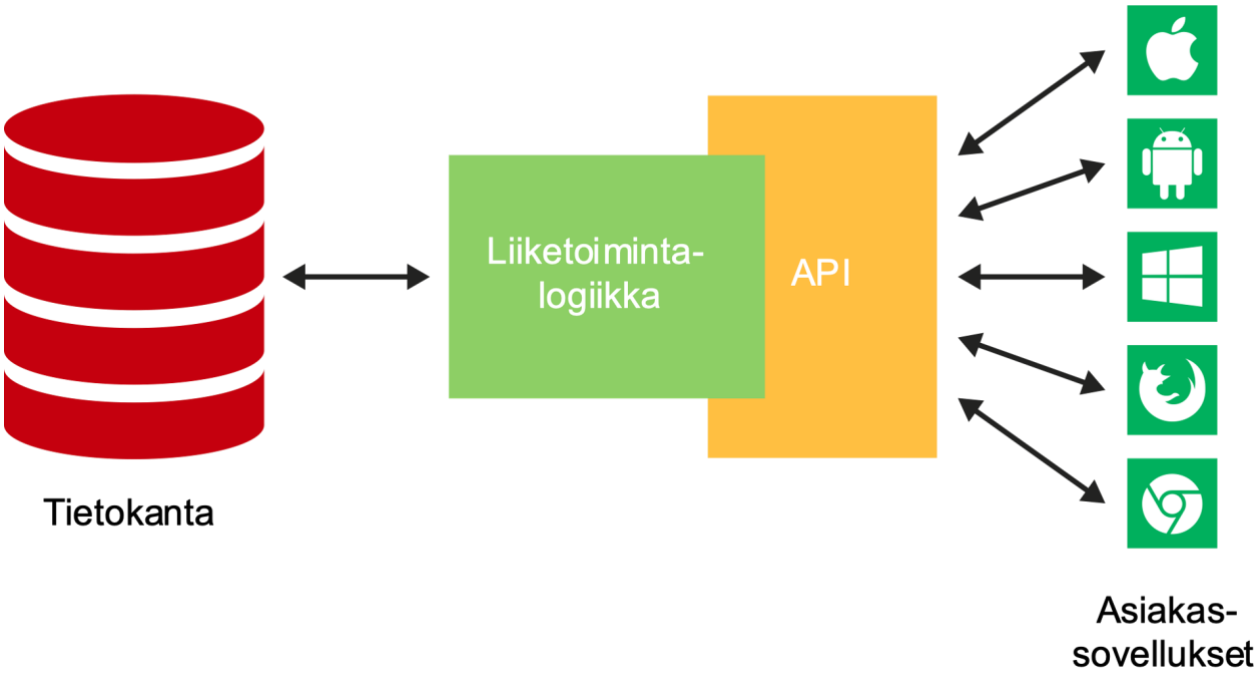
Ohjelmointirajapinnat voivat olla yrityksen avain taloudelliseen menestykseen. Jo vuonna 2015 Harvard Business Review kertoo artikkelissaan, että Salesforce.com tuloista 50 % tulee rajapintojen kautta, sillä yritys tarjoaa markkinapaikan yhteistyökumppanien luomille sovelluksille. Verkkomatkatoimisto Expedia.com saa 90 % tuloistaan rajapintojen avulla. Verkkohuutokauppa eBay puolestaan tarjoaa rajapintansa kautta muille verkkosivuille mahdollisuuden listata huutokauppakohteita verkkosivuillaan. Rajapintojen kautta eBay saa 60 % tuloistaan ja lisää näkyvyyttä huutokaupatuille tuotteille. (Iyer & Subramaniam 2015) Vuonna 2018 salatusta HTTPS-verkkoliikenteestä 83 % muodostui rajapintakutsuista. Määrä oli kasvanut merkittävästi vuoden 2014 lukemasta, joka oli 47 %. (Akamai 2019, 13)

Mikä rajapinta sitten on? Open Knowledge Finland Ry:n ylläpitämän avoimen rajapinnan määritelmän mukaan ohjelmointirajapinta spesifioi miten tietoja tai palveluita tarjotaan toisille sovelluksille tai tietojärjestelmille. Rajapinnasta yleisesti käytettävä API-lyhenne tulee englannin kielen sanoista Application Programming Interface. Rajapinnasta voidaan tarjoilla palvelun sisältämää dataa toisille järjestelmille. Tällöin rajapintaa kutsutaan datarajapinnaksi. Rajapinta voi tarjota käyttäjilleen laskenta-algoritmeja tai sen kautta voidaan muuttaa järjestelmän tietoja. Silloin rajapintaa voidaan kutsua toiminnalliseksi rajapinnaksi. (Open Knowledge Finland ry. Open API -työryhmä. 2014)

Ohjelmistorajapintoja on monenlaisissa yhteyksissä. Käyttöjärjestelmän rajapinnoilla on monipuolisia ominaisuuksia, jolla voidaan kontrolloida ohjelmistoa suorittavaa tietokonetta ja sen oheislaitteita, tai vaikka käyttöliittymän toiminnallisuuksia, kuten ikkunoiden vierityspalkkeja tai mobiililaitteilla kosketusnäytön käyttöä. Kaikkein yksinkertaisimpia rajapintoja edustavat web-rajapinnat voivat olla tarjota vain muutamia toimintoja yksittäisten tietojen hakemiseen. (Christensson 2016)

Ohjelmointirajapinnan käsite voidaan määritellä myös paljon yksityiskohtaisemmin ja teknisemmin. IEEE-standardissa oppimisteknologialle kerrotaan, että rajapinnan avulla sovellus voi käyttää ohjelmistoissa yleisesti käytettyjä kutsuja, funktioita, keskeytyksiä tai dataformaatteja päästäkseen käsiksi verkkopalveluihin, laitteisiin tai käyttöjärjestelmään (IEEE Standardi 1484.11.2 2020). Määritelmää voidaan laajentaa niin, että rajapinnan toiminta käsittää myös protokollat ja objektit, joita voidaan käyttää ohjelmistojen luomiseen tai vuorovaikutukseen ulkoisen järjestelmän kanssa. Rajapinta tarjoaa siis ohjelmistokehittäjille toimintojen suorittamiseen vakioituja komentoja, jotka helpottavat kehitystyötä. (Christensson 2016)

Rajapinta-arkkitehtuuri määrittelee, mitä tietoa rajapinnan kautta voi liikuttaa. Lisäksi se määrittää tavan, miten sovellus voi käyttää tätä informaatiota (Juviler 2021). Rajapintoja voidaan jaotella eri teknologioiden mukaan tai rajapinnan käyttöoikeuksien mukaan. Esimerkkejä eri web-rajapintojen teknologioista ovat REST ja SOAP. Käyttöoikeuksien mukaan rajapinnat jaotellaan yksityisiin, yhteistyökumppani ja julkinen rajapinta. Julkiset rajapinnat voidaan vielä jakaa vapaisiin ja ilmaisiin rajapintoihin ja kaupallisin rajapintoihin. (Maayan 2021)



Kuva 1. Rajapinnan havainnekuva (mukaillen Walker 2022a)

2.1.1 Web-rajapinnat ja niiden hyödyt

Web-rajapinta on siis ohjelmointirajapintaa täsmentävä käsite, joka kertoo rajapinnan olevan saatavilla verkosta HTTP-protokollan kautta. Niitä käytetään verkkopalvelimien ja verkkoselainten väliseen viestintään. Web-rajapinnat ovat riippumattomia ohjelmointikielistä ja niitä voidaan kehittää millä tahansa ohjelmointikielellä. Web-rajapinnoissa voidaan käyttää monia erilaisia tekniikoita kuten esimerkiksi SOAP, XML-RPC, JSON-RPC ja REST. (Raj & Subramanian 2019, luku Discussing the web API)

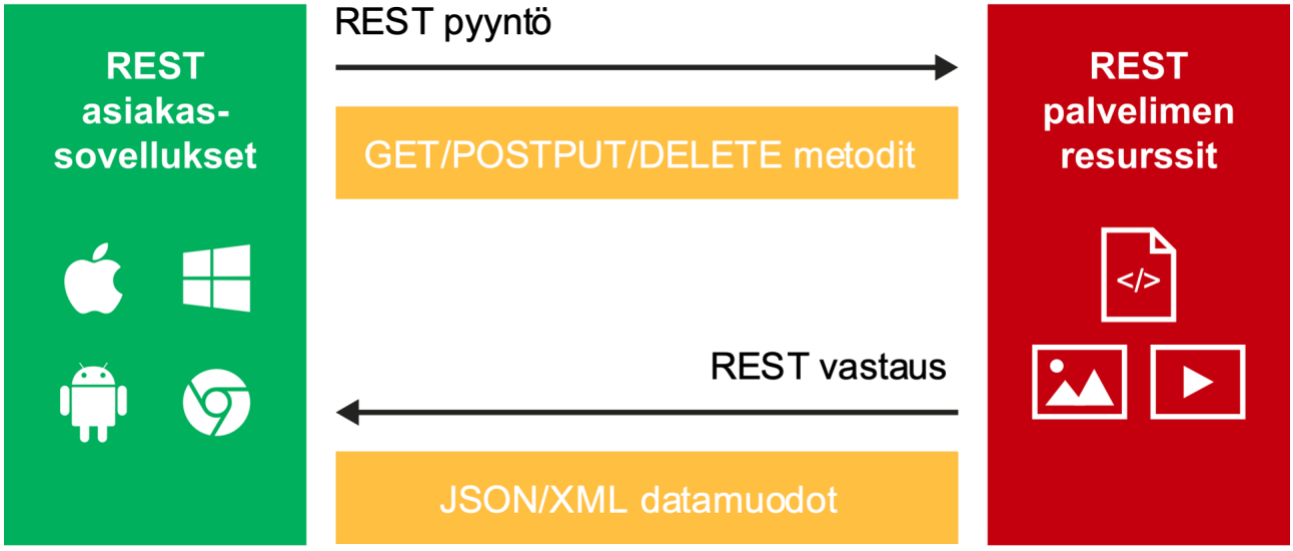
Rajapintojen hyötyjä voidaan tarkastella monestakin lähtökohdasta. Yksi näkökulma on rajapinnan käyttäjilleen tuoma lisäarvo, jolla on viisi ominaisuutta. Ensimmäinen ominaisuus on, että rajapinnan standardoidut toimintatavat vähentävät monimutkaisuuden kokemusta. Toiseksi rajapinnat tarjoavat paremman ja avoimemman pääsyn tietoon. Kolmanneksi ne antavat mahdollisuuksia vaikuttaa sisältöjen kehitykseen, koska rajapinnat tuovat uuden kanavan datan tarjoajille. Neljänneksi rajapinnat alentavat koettua riskiä, koska rajapinta on mahdollista vaihtaa

suhteellisen helposti. Lisäksi ne parantavat näkyvyyttä, innostavat innovaatioihin ja havainnollistavat hyötyjä, joita voidaan luoda avoimella datalla. (Moilanen, Niinioja, Seppänen ja Honkanen 2018, 33)

Ohjelmointirajapinnat nopeuttavat innovointia, koska ne mahdollistavat ohjelmistokehityksen jo olemassa olevan datan ja valmiiksi rakennettujen toimintojen perusteella. Ohjelmointirajapinnan avulla tietoja ja toimintoja voidaan välittää turvallisesti ja nopeasti. Rajapinnat tarjoavat selkeän määritelmän ja yksinkertaisen määritelmän ohjelmistojen vuorovaikutuksesta. Näin rajapinnat nopeuttavat ohjelmistokehitystä ja muuttavat tapaa, jolla yritykset rakentavat ohjelmistoja. (Poetker 2022)

2.1.2 REST -rajapinta-arkkitehtuuri

REST (Representational State Transfer) on yleisimmin käytettyä web- rajapinta-arkkitehtuuri (Simpson 2022), joka tarjoaa hajautetuille tietojärjestelmille viitekehyksen kommunikointiin HTTP-protokollaa hyödyntäen. REST-arkkitehtuurissa voidaan käyttää yleisistä käytössä olevia http-metodeja kuten GET, PUT, POST ja DELETE. Näiden avulla REST-arkkitehtuuri mahdollistaa asiakassovelluksille pääsyn haluttuihin resursseihin URL-osoitteen ja id-tunnuksien perusteella. REST voi käyttää useita datamuotoja tulosten näyttämiseen esimerkiksi JSON, XML tai tavallista tekstiä. (Raj & Subramanian 2019, luku Beginning with REST ja Characterizing the REST architecture style) Kuvassa 2. on kuvattu REST-rajapinnan toimintaa.



Kuva 2. REST-rajapinnan toiminta (mukaillen Altexsoft 2022)

REST-arkkitehtuurin tunnusomaisia piirteitä ovat asiakassovelluksen ja palvelimen välinen kommunikaatio, jossa asiakassovellus voi olla mikä tahansa pyynnon lähettävä sovellus. Palvelin taas ylläpitää REST-palvelua, joka tarjoaa asiakassovellukselle toivotut toiminnallisuudet ja

liiketoimintalogiikan. REST-arkkitehtuurissa palvelin on tilaton eli se ei tallenna istuntotietoja. Sen vuoksi asiakassovelluksen täytyy lähettää pyynnön mukana kaikki vaaditut tiedot vastauksen saamiseksi. REST-palveluissa on olennaista huomioida asiakassovelluksien välimuistin käyttäminen. Asiakassovellukset tallentavat tietoja välimuistiin toiminnan nopeuttamiseksi ja verkkoliikenteen vähentämiseksi. Tämän vuoksi samanlaisen pyynnön vastaus voidaan tarjoilla asiakassovelluksen välimuistista. REST on myös kerroksellinen järjestelmä, jossa voi olla eri moduuleja palvelimen ja asiakassovelluksen välissä. Tämän tarkoituksena on vähentää monimutkaisuutta yhä laajemmissa järjestelmissä. (Raj & Subramanian 2019, alaluku Characterizing the REST architecture style)

REST-rajapinnat helpottavat uusien sovellusten integroimista olemassa oleviin järjestelmiin. REST-arkkitehtuurin käyttäminen nopeuttaa ohjelmistokehitystä, kun jokaista toimintoa ei tarvitse kirjoittaa tyhjästä, vaan voidaan hyödyntää olemassa olevaa koodia. Innovaatiot helpottuvat, kun rajapintojen avulla uusia palveluita voidaan ottaa käyttöön nopeasti. Näin voidaan reagoida ripeästi toimialan muutoksiin. Rajapinnat mahdollistavat toiminnan laajentamisen eri alustoille ja näin voidaan vastata asiakkaiden tarpeisiin. Rajapinnat ovat yhdyskäytävä kahden sovelluksen välillä. Rajapinnat siis helpottavat järjestelmien ylläpitoa, kun toisen järjestelmän muutokset eivät vaikuta toisen osapuoleen. (Amazon AWS 2022)

Asiaa voidaan tarkastella myös ohjelmistokehittäjän näkökulmasta, jolloin huomataan, että REST-arkkitehtuurin käyttämisestä on monia hyötyjä. REST-arkkitehtuurilla on mahdollista parantaa kehittäjätiimin tuottavuutta, koska REST-arkkitehtuuria on helppo ymmärtää. Sen vuoksi REST-rajapintoja myös helppo toteuttaa. REST-rajapinnoilla istunnot eivät vie palvelimen resursseja. Se puolestaan helpottaa sovellusten skaalaamista tarpeen mukaan. Toinen skaalattavuuteen vaikuttava seikka on se, että REST-rajapintojen tyypillisesti käyttämä JSON-datamuoto on kooltaan pienempi ja nopeampi jäsentää kuin XML. REST-rajapintojen avulla voidaan lyhentää palvelimien vastausaikaa, koska GET- ja POST-pyyntöjä voidaan tallentaa välimuistiin. Tämä on kriittistä palvelujen suorituskyvyn kannalta. Lisäksi REST-rajapinnat ovat joustavia ja muutoksia on helppo tehdä ilman, että asiakassovelluksia täytyy muuttaa. (Erinç 2020)

Vaikka REST-rajapinnat ovat hyvin suosittuja, niin web-rajapintojen toteutukseen on paljon muitakin vaihtoehtoja. GraphQL on rajapinnan kyselykieli, jonka avulla tehdään kyselyjä palvelimelle ennalta määriteltyihin tietotyyppeihin. GraphQL rajapinnassa jokaiselle tietotyypille luodaan omat funktiokyselyt, joka palauttaa halutun arvon. GraphQL palvelun perusajatus on, että se vastaanottaa kyselyitä, joissa pyydetään tiettyjä resursseja. Ensin palvelu tarkistaa, että kyselyssä olevat tietotyypit ovat määritellyt. Tämän jälkeen palvelu suorittaa tarjotut funktiokyselyt ja tuottaa vastauksen esimerkiksi JSON-muodossa. (The GraphQL Foundation 2022a) Erotuksena

REST-rajapintoihin GraphQL palvelu vastaa vain yhdessä osoitteessa, joka tuottaa kaikki ominaisuudet (The GraphQL Foundation 2022b).

Ennen REST-arkkitehtuurin syntyä monet rajapinnat toteutettiin SOAP on protokollaa hyödyntäen. Näiden kahden merkittävin ero onkin se, että SOAP on protokolla, kun taas REST on arkkitehtuurinen valinta. REST-rajapinnassa voidaan siis hyödyntää SOAP protokollaa, mutta SOAP ei voi hyödyntää REST-arkkitehtuuria. (Walker 2023b) SOAP mahdollistaa standardisoidun protokollan sovellusten jakamille viesteille, joiden pohjana vahvasti standardisoitu XML-merkkauskieli. SOAP rajapinnan voidaan sanoa olevan XML-sovellus. SOAP:n perusajatuksena on antaa raamit sovellusten viestinnälle. Vaikka XML on standardisoidu merkkauskieli, niin saman asian voi esittää XML:ssä eri tavoin. SOAP puolestaan määrittelee yhtenäisen tavan strukturoida XML-viestejä. Näin kaksi sovellusta voivat jakaa tietoa XML-viestein riippumatta käyttöjärjestelmästä, ohjelmointikielestä tai mistä tahansa muusta teknisestä ominaisuudesta. (Kulchenko, Snell & Tidwell 2001, alaluku SOAP and XML)

2.1.3 Rajapintojen datamuodot

Data voi liikkua monessa eri muodossa rajapintojen välityksellä. Datamuoto onkin yksi tärkeimmistä tekijöistä, joka tulee ottaa huomioon ohjelmistojen suunnittelussa, toteutuksessa ja ylläpidossa. Käyttötarkoitukseen sopiva dataformaatti on tärkeä valinta, joka voi erottaa hyödyllisen ja tehokkaan rajapinnan vajaakäyttöisestä rajapinnasta, jonka täyttä potentiaalia ei pystytä hyödyntämään. Web-rajapinnoilla tyypillisimmin käytettyjä dataformaatteja ovat JSON, XML, ja YAML. (Sandoval 2016)

JSON on yksi yleisimmistä käytetyistä formaateista etenkin REST-rajapinnoissa. 69 % verkkoliikenteestä oli JSON-muotoista dataa vuonna 2018. Vuonna 2014 % JSON osuus oli 26 %. Samalla aikajänteellä XML käyttö on vähentynyt vain 17 % aiemmasta 54 %. Suurin osa 66 % JSON liikenteestä tulee älypuhelimuista, sovelluksista tai muista sulautetuista järjestelmistä kuten pelikonsoleista ja älytelevisoista. (Akamai 2019, 13–16)

JSON perustuu joulukuussa 1999 esiteltyyn JavaScriptin ECMA-262 ohjelmointikielistandardiin, jonka osaksi JSON otettiin vuonna 2009 (ECMA-262 13 versio kohta Johdanto 2022). JSON on yleinen, web-rajapinta-arkkitehtuureissa käytetty datamuoto, jonka lyhenne tulee sanoista JavaScript Object Notation. JSON on helppolukuinen ihmisille, mutta samaan aikaan tietokoneiden on tehokasta jäsennellä ja luoda JSON-tiedostoja. JSON on tekstiformaatti, joka täysin riippumaton ohjelmointikielestä, mutta nojaa vahvasti C-kieliperheen käytäntöihin. (Introducing JSON)

JSON rakentuu kahdelle periaatteelle, jotka ovat nykyaikaisissa ohjelmointikielissä yleisesti tunnettuja, universaaleja tietorakenteita. JSON-datamuodossa tieto on jäsennelty nimen ja arvon

muodostamiin pareihin. Useissa ohjelmointikielissä vastaava rakenne on toteutettu objekteina. Nämä nimestä ja arvosta rakentuvat objektit muodostavat puolestaan järjestelyn listan tai kokoelman, jotka ovat monissa ohjelmointikielissä toteutettu taulukkoina. Näitä JSON-datamuodossa esiintyviä datarakenteita tukevat lähes kaikki ohjelmointikielet muodossa tai toisessa. Tämä tekee JSON:sta ohjelmistokehittäjälle käytännöllisen tiedonsiirtomuodon. JSON:ssa hyväksyttäviä arvoja ovat objektit, taulukot, merkkijonot, numerot, totuusarvot tai tyhjät null-arvot. (Introducing JSON)

XML on toinen laajasti tiedonsiirrossa käytetty tiedostoformaatti. XML tulee englannin kielen sanoista Extensible Markup Language ja se on nimensä mukaisesti merkintäkieli kuten esimerkiksi HTML-kieli. HTML-kieleen verrattuna XML-syntaksi on hyvin tiukka. XML on World Wide Web Consortium (W3C) suositus, joka pohjaa toiseen vanhampaan tiedonsiirtoformaattiin nimeltä SGML (ISO 8879). (World Wide Web Consortium (W3C) 2015)

World Wide Web Consortium kertoo, että XML tiedostoissa on tiedonsiirtoon monia hyviä ominaisuuksia. Merkintäkielen ansiosta virheet ovat helppo havaita, sillä kaikkien tagien lopputunnisteet ovat pakollisia. Ihmisten on myös mahdollista lukea tietoja ja havaita virheitä tagien ja niiden attribuuttien avulla. XML-tiedostoja voidaan myös lukea useilla työkaluilla, vaikka ne ovat joskus sidottuja tiettyyn XML-merkintään. (2015)

JSON ja XML eroavat toisistaan monellakin tavalla. JSON objekteilla on aina tyyppi, kun taas XML data on tyypitöntä. JSON ei tue nimiavaruutta toisin kuin XML. JSON ei tarjoa keinoja datan näyttämiseen, kun taas XML on merkintäkieli kuten esimerkiksi HTML. JSON merkistökoodaus on aina UTF-8 kun taas XML tukee useita eri merkistökoodauksia. (Walker 2022c)

2.2 Datan validointi

Rajapinnoilla datan ja käyttäjäsyötteiden validointi on välttämätöntä lähes jokaisessa sovelluksessa. Käyttäjät syöttävät järjestelmään tietoa, joka välitetään rajapinnan kautta tallennettavaksi tietokantaan. Lisäksi sovellukset välittävät dataa keskenään erilaisten rajapintojen avulla. Oli käyttötapaus mikä tahansa, niin välitettävä data tulisi aina validoida. Näin voidaan vähentää tietoturvaongelmia sekä datan eheys ja laatu ovat helpompia varmistaa.

Tietojen validointi on prosessi, jossa verrataan syötettyjä tietoja ennalta määriteltyihin sääntöihin. Näin varmistetaan, että syötetyt tiedot ovat vaatimusten mukaisia. Validointi on sarja tarkastuksia, jotka vaihtelevat yksinkertaisista hyvin edistyneisiin tarkastuksiin, joihin sisältyy strukturoituja ehdollisia tarkastuksia. Validoinnin tarkoituksena on varmistaa, että data on puhdasta, käyttökelpoista ja täsmällistä. Validoimaton data voi johtaa epätarkkoihin tuloksiin, sovellusten kaatumisiin ja muihin vaikeisiin ongelmiin. (Phoenix 2022)

Ensinnäkin datan validointi on syötettävän datan syntaksin tarkistamista. Lisäksi datan täytyy olla järjestelmän vaatimusten mukaisesti muotoiltua. Syötettävän datan tyyppi tarkistetaan odotetun tyypin mukaisesti. Esimerkiksi numeroita sisältävään kenttään ei voi syöttää kirjaimia. Tiedot on myös tarkistettava, että ne sisältävät vain arvoja, jotka ovat odotetulta arvoalueelta. (Phoenix 2022) Yksi esimerkki arvoalueesta voisi olla ISO 3166 -standardin mukaiset maakoodit.

Toisaalta datan validoinnin voi määritellä myös sen tarkoituksen mukaan. Datan validointi voi perustua liiketoiminnansääntöihin, syntaksisääntöihin tai semanttisiin sääntöihin. Liiketoiminnan sääntö voi olla esimerkiksi se, että jokin tieto on pakollinen. Syntaksi säännöt taas määrittelevät esimerkiksi tietotyypin ja pituuden. Semanttiset säännöt varmistavat, että tiedon arvot ovat kelvollisia. Esimerkkinä voisi olla se, että BIC-koodi eli pankin yksilöivä tunniste on oikeassa muodossa. (DTCC 2022)

2.2.1 Validoinnin hyödyt

Sovelluksessa olevan syötteen validoimisen tavoitteet ovat selkeät: sen tarkoituksena on varmistaa, että tietojärjestelmän työnkulkuun pääsee vain oikean muotoista data. Validointi estää virheellisen datan syöttämisen tietokantaan ja estää jatkossa ohjelmiston komponenttien toimintahäiriöitä. Kaikki mahdollisesti epäluotettavista lähteistä tulevat syötteet on tarkistettava. Tietoturvan näkökulmasta validointia ei ole ensisijainen keino estää XSS- ja SQL-injektiota sekä muita hyökkäyksiä. Oikein toteutettuna validointi voi kuitenkin vähentää niiden haitallista vaikutusta. (OWASP Cheat Sheet Series Team 2021)

Datan validointi varmistaa, että lähetettävä tai vastaanotettava tieto on virheetöntä ja eheää. Oikein määriteltynä validointi varmistaa, että data on käyttötarkoituksen mukaista. Datan validointi auttaa myös useiden eri tietolähteiden kanssa, kun voidaan varmistaa, ettei eri lähteistä tulleen datan välillä ei ole eroja tai virheitä. Tietojen validointi säästää aikaa, kun järjestelmään ei tarvitse tehdä muutoksia elleivät sen vaatimukset muutu. Lisäksi validointi on ongelmien syntymistä estävää ennakoivaa toimintaa, koska tietojen oikeellisuus varmistetaan ennen niiden käyttöä. (Phoenix 2022)

2.2.2 Validointiskeemat

Validoinnin yhteydessä mainitaan usein skeemat. Ne määrittelevät, kuinka data on järjestetty. Skeema on siis eräänlainen ääriviiva, kaavio tai malli, jota käytetään kuvaamaan erityyppisten tietojen rakennetta. Kaksi yleistä esimerkkiä ovat tietokanta- ja XML-skeemat. (Christensson 2018)

JSON skeeman verkkosivuilla kerrotaan, että JSON skeema on sanasto, jonka avulla voidaan kuvailla JSON muotoista dataa ja sen vaatimuksia. Niiden perusteella voidaan validoida JSON-

tiedostoja ja niiden sisältämää dataa. JSON skeemat ovet itsessään myös JSON muotoisia. JSON skeemat tarjoavat dataformaateille dokumentaation, jota on helppo lukea ihmisten ja tietokoneiden näkökulmasta. (OpenJS Foundation 2022a)

Jos ajatellaan validointia ylipäätänsä, niin on mahdollista validoida JSON dokumentin syntaksi ilman skeeman käyttämistä. Näin voidaan validoida vain se, että JSON dokumentti on muotoiltu oikein. Usein on tärkeää myös validoida datan sisältöä ja laatua. JSON skeeman avulla datan sisällön tarkistava semanttinen validointi on mahdollista. Näin voidaan tarkistaa esimerkiksi, että puhelinnumerot ja päivämäärät ovat oikeassa formaatissa tai että JSON sisältää vain tarvittavat kentät ilman ylimäistä dataa. (Marrs 2017, alaluku 5. JSON Schema)

Toisaalta JSON skeema on itsessään JSON muotoista tietoa ja se ei voi sisältää mielivaltaista koodia. Siksi JSON skeeman avulla ei voida ilmaista kaikkia tietoelementtien välisisä suhteita. Hyvin monimutkaisten tietojen validointi on todennäköisesti toteutettava jollakin ohjelmointikielellä. (Droettboom 2020) Monimutkaisimmissa tapauksissa semanttiseen validointiin JSON skeeman avulla on siis suhtauduttava varauksella. Yksinkertaisemman tiedon kohdalla JSON skeemalla voidaan toteuttaa liiketoiminta- ja syntaksisäännöt sekä semanttisiin sääntöihin perustuva validointi.

Data validoinnissa JSON skeemojen tarkoitus on samantyyppinen kuin XML-dokumenttien kanssa käytettyjen XML skeemojen. Merkittävimmät erot ovat seuraavat: JSON-dokumentti ei viittaa tiettyyn JSON skeemaan vaan sovelluksen tulee validoida JSON skeeman perusteella. JSON skeemoilla ei ole nimiavaruutta ja tiedostoilla on .json-tiedostopääte. (Marrs 2017, alaluku 5. JSON Schema)

JavaScript tai TypeScript -projekteissa suosituin validointikirjasto Ajv JSON schema validator hyödyntää nimensä mukaisesti JSON skeemoja. Kirjaston suosiosta kertoo se, että Npm-sivustolla sillä on 76,7 miljoonaa viikoittaista latausta. Toiseksi suosituimmalla kirjastolla JSON Schema on 21.7 miljoonaa viikoittaista latausta. (Npm, Inc 2022) Muille ohjelmointikielille on olemassa vastaavia toteutuksia ja melko kattava lista löytyy JSON skeeman verkkosivuilta (OpenJS Foundation 2022b).

2.3 Ohjelmistokirjastot

Edellisessä kappaleessa mainittu Ajv JSON schema validator on validointiin käytettävä ohjelmistokirjasto, jonka tarkoitus on helpottaa ohjelmistokehittäjien työtä tarjoamalla heille valmiiksi kirjoitettuja toimintoja. Validointi ei ole kuitenkaan ainoa tilanne, jossa voidaan hyödyntää valmiiksi kirjoittua ohjelmakoodia. Ohjelmistokirjastoja onkin hyvin moneen eri tarkoitukseen. Seuraavaksi käsittelemme mitä ohjelmistokirjastot oikein ovat.

Ohjelmistokirjasto on kokoelma uudelleenkäytettävää ohjelmakoodia, jota hyödynnetään ohjelmistojen ja sovellusten kehittämiseen. Se on suunniteltu auttamaan sekä ohjelmoijaa että ohjelmointikielen kääntäjää ohjelmistojen rakentamisessa ja suorittamisessa. Ohjelmistokirjasto koostuu yleensä ennalta kirjoitetusta koodista, luokista, funktioista, konfiguraatiotiedoista ja muista osista. (Janssen & Janssen 2016)

Ohjelmistokirjaston ja ohjelmistokehyksen merkittävin ero on ohjelman hallinta suorituksen aikana. Ohjelmistokehittäjä kutsuu ohjelmakoodissaan ohjelmistokirjaston luokkia ja funktioita. Ohjelmistokirjaston kohdalla kehittäjä määrittelee missä ja milloin ohjelmistokirjaston ominaisuuksia kutsutaan. Ohjelmistokehyksen kohdalla tilanne on toinen ja ohjelmistokehyksen metodit kutsuvat suorituksen aikana kehittäjän kirjoittamia funktioita. Ilmiötä kutsutaan käänteiseksi hallinnaksi. (Fowler 2005)

IT-sopimusehtoja käsittelevässä IT2022 – Käytännön käsikirjassa pohditaan avoimen lähdekoodin nopeuttavan tuotekehityssykliä ja alentavat tutkimus- ja tuotekehityskustannuksia. Avoimen lähdekoodin ohjelmistot nähdään myös laadukkaina, niiden perustuessa laajan kehittäjäyhteisön osaamiseen. (Erlund 2022, 216–217) IT2020 sopimusehdoissa vapaalla lähdekoodilla viitataan juuri ohjelmistokirjastoihin ja ohjelmistokehyksiin.

Datan validointiin käytettäviä ohjelmistokirjastoja on lukuisia. Node.js:n paketinhallintaohjelmiston Npm:n verkkosivuilta löytyy hakusanalla ”validation” yli 10000 hakutulosta (Npm, Inc 2022). JavaScript- tai TypeScript-projektissa on siis hyvin paljon valinnanvaraa. Muilla ohjelmointikielillä on luonnollisesti omat ohjelmistokirjastonsa validointiin. Sopivan kirjaston valitseminen on merkittävässä roolissa validoinnin toteuttamisessa ohjelmistoprojektissa.

2.3.1 Ohjelmistokirjastojen valintaperusteet

Ohjelmistokirjastojen valintaperusteita voidaan jaotella teknisiin, inhimillisiin ja taloudellisiin tekijöihin. Tekniset tekijät sisältävät itse kehitettävään ohjelmistoon liittyvät tekijät ja kirjaston toiminnallisuuteen, laatuun ja ylläpitoon liittyviä tekijöitä. Inhimilliset tekijät ovat sidosryhmiin, organisaatioon, yksilöön ja yhteisöön liittyviä tekijöitä. Taloudelliset tekijät ovat kokonaisomistuskustannukset (TCO) ja riskit. (Vargas, Aniche, Treude, Bruntink & Gousios 2020)

Teknisiä valintaperusteita tarkasteltaessa itse ohjelmistoon liittyvät tekijät ovat sen teknisen toteutuksen aiheuttamia rajoitteita. Ohjelmiston kehittämisessä täytyy ottaa huomioon riippuvuudet toisiin ohjelmistoihin. Esimerkiksi kehitetäänkö täysin uutta ratkaisua vai olemassa olevaa ohjelmistoa. Toiminnalliset tekijät kattavat ohjelmistokirjaston koon ja monimutkaisuuden sekä sen sopivuuden käyttötarkoitukseen. Laatuun liittyvät tekijät ovat kirjaston sopiminen ohjelmistoarkkitehtuuriin, käytettävyys, dokumentaatio, suorituskyky ja testauksen kattavuus.

Kirjaston ylläpitoon liittyvät tekijät tarkoittavat kirjaston ylläpidon aktiivisuutta, kypsyyttä ja vakautta sekä julkaisujen aikaväliä. (Vargas ym. 2020)

Inhimillisistä tekijöistä sidosryhmät tarkoittavat esimerkiksi asiakkaita, kehitystiimiä, muita ohjelmiston parissa työskenteleviä tiimejä ja tuoteomistajaa. Sidosryhmillä kuten kehitystiimillä ja yksittäisellä jäsenellä on erittäin merkittävä vaikutus ohjelmistokirjaston valintaan. Toinen inhimillinen tekijä on yksittäisen ohjelmistokehittäjän näkemys. Kolmas inhimillinen tekijä on organisaation vaikutus. Organisaation tekemät teknologiset linjaukset, kulttuuri, käytänteet ja toimiala vaikuttavat merkittävästi ohjelmistokirjaston valintaan. Inhimillisiin tekijöihin lasketaan myös ohjelmistokirjaston ympärillä oleva yhteisö, joka voidaan jakaa kirjastoa hyödyntäviin käyttäjiin ja kirjastoa ylläpitäviin sekä teknistä tukea antaviin ihmisiin. (Vargas ym. 2020)

Valintaperusteiden taloudellisista tekijöistä ensimmäinen on kokonaisomistuskustannukset TCO, joka tulee englannin kielen sanoista Total Cost of Ownership. Tällä tarkoitetaan kaikkia asioita, jotka liittyvät kehityskustannuksiin, lisensointiin ja kaupallisten ohjelmistokirjastojen hintaan. Kehityskustannuksissa punnitaan, että kuinka paljon kirjaston käyttö nopeuttaa kehitysprosessia. Lisensointi vaikuttaa olennaisesti ohjelmiston hyödyntämiseen kaupallisissa tarkoituksissa. Toinen taloudellinen tekijä on ohjelmistokirjastoon liittyvät riskit, kuten ohjelmistokirjaston saatavuus tulevaisuudessa ja käyttöönoton hinta. Riskiä pienentävänä tekijänä mainitaan esimerkiksi ohjelmistokirjastolla oleva tunnetun yhteisön tuki. (Vargas ym. 2020)

Teknisistä ohjelmistokirjaston valintaperusteista merkityksellisimpiä tekijöitä olivat käytettävyys, dokumentaatio, ohjelmistokirjaston kypsyys ja vakaus sekä aktiivinen ylläpito. Inhimillisistä tekijöistä merkityksellisimpiä olivat ohjelmistokirjaston ympärillä olevan yhteisön aktiivisyys ja suosio. Taloudellisista tekijöistä lisensoinnilla oli suurin merkitys. (Vargas ym. 2020)

Valintaperusteita voidaan tarkastelle myös toisenlaisesta näkökulmasta. NPMCompare on verkkosivusto, joka vertailee Node.js:n pakettimanagerin Npm:n jakelemia JavaScript ja TypeScript ohjelmistokirjastoja. Sivusto vertailee ja pisteyttää ohjelmistokirjastoja 16 eri perusteella. Vertailun tekijöitä ovat lisenssi, luontipäivämäärä, viimeisin muokkaus, versioita, ylläpitäjät, riippuvuudet ja wikituen tarjoaminen. GitHubin tarjoamista tiedoista vertaillaan avoimia ongelmia ja vetopyyntöjä, haarojen, tähtien ja seuraajien lukumäärään. Lisäksi vertailussa huomioidaan ohjelmistokirjaston päivittäiset, viikoittaiset ja kuukausittaiset latausmäärät. (NPMCompare 2022).

NPMCompare on käytännöllinen työkalu JavaScript ja TypeScript ohjelmistokirjaston valinnassa. Monet palvelussa vertailtavat tekijät ovat hyviä valintaperusteita millä tahnansa ohjelmointikielellä kirjoitetulle ohjelmistokirjastolle. Esimerkiksi GitHubin avoimet ongelmat kielivät ohjelmistokirjaston luotettavuudesta. Varsinkin, jos GitHubissa on paljon avoimia ongelmia ja kirjastoa ei ole päivitetty

pitkään aikaan. Myös ohjelmistokirjaston ylläpitoon ja tukeen osallistuvat ihmiset ovat tärkeitä mille tahansa kirjastolle.

3 Kyselytutkimus ohjelmistoalan ammattilaisille

Tässä opinnäytetyössä esiteltävän tutkimuksen tavoite on ollut saada selville mitkä seikat vaikuttavat siihen, kuinka web-rajapintasovelluksia kehittävät ammattilaiset valitsevat ohjelmistokirjastoja. Tarkoituksena on löytää käytännön kehitystyöhön sovellettavat perusteet ohjelmistokirjaston valinnalle ja tutkia eri valintaperusteita yleisellä tasolla.

Tutkimuksessa pyritään selvittämään, miten REST-arkkitehtuuri ja JSON datamuoto vaikuttavat validointikirjaston valintaan. Onko validointikirjaston valinnassa joitakin erityisvaatimuksia vai ovatko valintaperusteet samoja kaikkien ohjelmistokirjastojen kohdalla?

3.1 Kohderyhmä

Tutkimuksen kohderyhmänä ovat ohjelmistoalalla työskentelet henkilöt tai ohjelmistotuotantoa opiskelevat henkilöt. Kohderyhmä on valittu sen perusteella, että alalla työskentelevillä on hyvä käytännön kokemus tutkittavaan ilmiöön. Yksittäisen ohjelmistokehittäjän vaikutus ohjelmistokirjaston valinnassa saattaa olla suuri, mutta valintaan vaikuttavat myös muut sidosryhmät. Sen vuoksi kohderyhmä on ohjelmistoalalla työskentelevät henkilöt eivätkä vain ohjelmistokehittäjille. Kohderyhmässä on siis tarkoitus olla edustettuna valintaan vaikuttavat sidosryhmät, kuten tuoteomistajat ja ohjelmistojen myyntityötä tekevät ihmiset. Potentiaalisia vastaajia tavoiteltiin sosiaalisen median alustoilla ja viestisovellusten suljetuissa ryhmissä.

3.2 Menetelmävalinnat

Tutkimusmenetelmänä on lomakekysely, joka on jaettu viiteen osioon. Ensimmäisessä osassa selvitetään vastaajien taustatietoja, kuten koulutustaustaa ja työkokemusta. Toisessa osassa käsitellään ohjelmistokirjastojen valintaa yleisellä tasolla riippumatta sen käyttötarkoituksesta. Kolmannessa osassa pyritään selvittämään validointiin käytettävän ohjelmistokirjaston valintaa ja sen erovaisuuksia verrattuna muiden ohjelmistokirjastojen valintaan. Neljännessä osassa keskitytään REST-rajapintojen vaikutetukseen validointiin käytettävän kirjaston valinnassa. Viimeisessä osassa samaa asiaa pohditaan JSON datamuodon näkökulmasta.

Kyselytutkimuksessa on 17 kysymystä. Monivalintakysymyksillä haetaan tietoa ohjelmistoalan ammattilaisten käyttämistä perusteista ja tekniikoita. Arviointiasteikko kysymyksiä on myös useaa eri tyyppiä. Yksinkertaisissa kysymyksissä on selkeä arviointiasteikko yhdestä tutkittavasta asiasta. Arviointiasteikkoa hyödynnettiin myös monimutkaisissa matriisikysymyksissä, joissa Likert-asteikot mittaavat erilaisten valintaperusteiden tärkeyttä kohderyhmän mielestä. Ilmiöt tutkimuskysymysten taustalla ovat melko monimutkaisia, joten tutkimuksessa on myös avoimia kysymyksiä. Niiden tarkoituksena on tuoda tutkittavaan asiaan syvyyttä. Avoimilla kysymyksillä on mahdollisuus

selvittää asioita, joita valmiissa vastausvaihtoehdoissa olisivat jääneet ehkä huomaamatta. Kyselytutkimus on kokonaisuudessaan tarkasteltavissa liitteessä 1.

Kyselytutkimuksen avulla saadaan siis kvantitatiivista ja kvalitatiivista aineistoa. Kvantitatiivisen aineiston avulla voidaan tehdä yleistettäviä päätelmiä ohjelmistokirjaston valintaan vaikuttavista tekijöistä ja niiden merkityksestä. Avoimista kysymyksistä saatava kvalitatiivinen aineisto auttaa ymmärtämään tutkimuskysymyksiin vaikuttavia tekijöitä tarkemmin ja koko ilmiötä laajemmin.

Kyselytutkimus laadittiin tutkimuskirjallisuuden tietojen valossa niin, että kysymyksillä saataisiin selkeitä vastauksia tutkimusongelmiin. Tutkimusinstrumentti validoitiin ennen käyttöä kohderyhmän edustajien avulla. Ennen julkaisua saadun palautteen perusteella tehtiin pieniä korjauksia ja tarkennuksia.

Kyselytutkimus toteutettiin ja aineisto kerättiin Webpropol-työkalulla. Kyselytutkimukseen oli mahdollista vastata 15.11. – 12.12.2022 välisenä aikana. Kyselytutkimuksen kohderyhmää tavoiteltiin omien sosiaalisten verkostojen avulla. Julkista nettilinkkiä jettiin sosiaalisen median palveluissa kuten LinkedIn sekä WhatsApp- ja Discord-ryhmissä. Sosiaalisen median palveluissa linkkiä jaettiin eteenpäin muutamia kertoja. Kysely oli anonyymi, joten vastaajia ei voinut tunnistaa kyselyn vastauksista.

3.3 Aineiston analysointi

Kvantitatiivista aineistoa analysoidaan yhteenveto- ja ristiintaulukoinnin kautta ja graafisia kuvaajia käytetään visualisoimaan tuloksia. Yksittäisten kysymysten keskilukuja kuten mediaania tai keskiarvoa voidaan käyttää hahmottamaan vastausten jakautumista eri vaihtoehdoille. Ohjelmistokirjaston valintaan vaikuttavien seikkojen merkitystä tarkastellaan myös vastausten hajonnan perusteella. Näin voidaan päätellä missä asioissa vastaajat ovat yksimielisiä. Näin voidaan hahmottaa mitkä asiat koetaan merkityksellisiksi valintaprosessissa.

Aineistoa suodatettiin myös vastaajien taustatietojen perusteella. Työkokemuksen vaikutusta vastauksiin vertailtiin tarkastelemalla kysymyksien vastauksia eri työkokemusvuosien perusteella. Muuttujana toimi siis työkokemus. Muita taustatietoja ei käytetty muuttujana tulosten analysoinnissa.

Kyselytutkimuksen laadullista aineistoa analysoitiin teemoittamalla aineistoa. Tutkimusaineistosta nousi esiin uusia asioita, joita ei ollut tunnistettu ja ennakoitu kyselytutkimusta laadittaessa. Sen vuoksi tarvittiin uusia teemoja, jotta avoimien kysymysten vastauksissa ilmi tulleet uudet asiat saivat tarpeeksi tilaa analysoinnissa. Toisaalta analyysin edistyessä huomattiin, että aineistossa toistui paljon asioita, jotka oli jo kyselytutkimusta laadittaessa tunnistettu. Oli siis luontevaa, että

jotkin ennalta tunnistetut valintaperusteet nousivat myös analyysin teemoiksi. Esimerkiksi ohjelmistokirjastojen dokumentaatioon viitattiin avoimien kysymysten vastauksissa useasti, joten tuki ja dokumentaatio oli teemana myös analysoinnissa. Kyselytutkimuksen vastaukset olivat melko lyhyitä, joten aineiston jakaminen teemoihin oli riittävä pohja tulosten analysointiin.

4 Tulokset

Kyselyyn vastasi yhteensä 22 henkilöä. Puolet kyselytutkimukseen vastaajista oli 31–40-vuotiaiden ikäryhmästä. 21–30-vuotiaiden ja 41–50-vuotiaiden ikäryhmästä oli molemmista 22,7 % vastaajista. Loput 4,6 % oli 51–60-vuotiaiden ikäryhmästä.

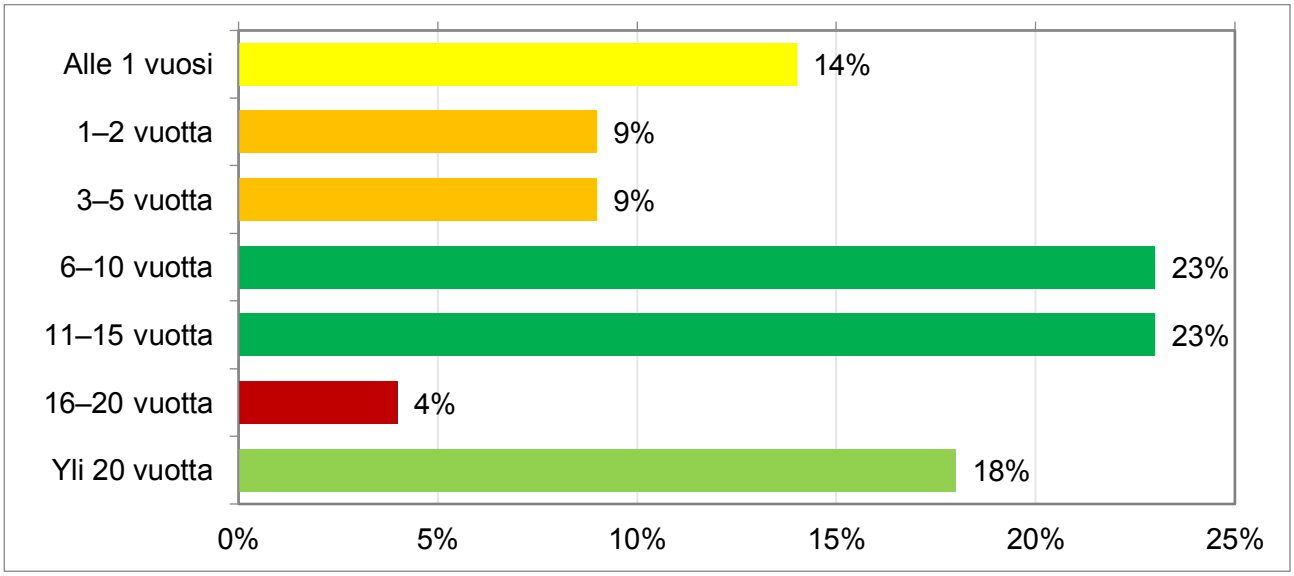
4.1 Vastaajien koulutustausta

Suurin osa 77,3 % vastaajista kertoi oman koulutusalan olevan tietojenkäsittely tai tietoliikenne (ICT). Loput 22,7 % vastaajista jakaantui tasaisesti humanistisille ja taidealoille, kauppa-, hallinto- ja oikeustieteisiin, luonnontieteisiin ja tekniikan aloille.

86,4 % vastaajista oli joko alemman korkeakouluasteen tai ylemmän korkeakouluasteen tutkinto, joten suurin osa vastaajista oli korkeakoulutettuja. Alempaa korkeakouluastetta tai ammatillista korkeakouluastetta edusti 45,5 % vastaajista ja 40,9 % vastaajista oli ylempää korkeakouluastetta. Näiden kahden vaihtoehdon välillä vastaajat jakaantuivat siis suhteellisen tasaisesti. Lopuilla vastaajista oli toisen asteen tai opistoasteen tutkinto.

4.2 Työkokemus

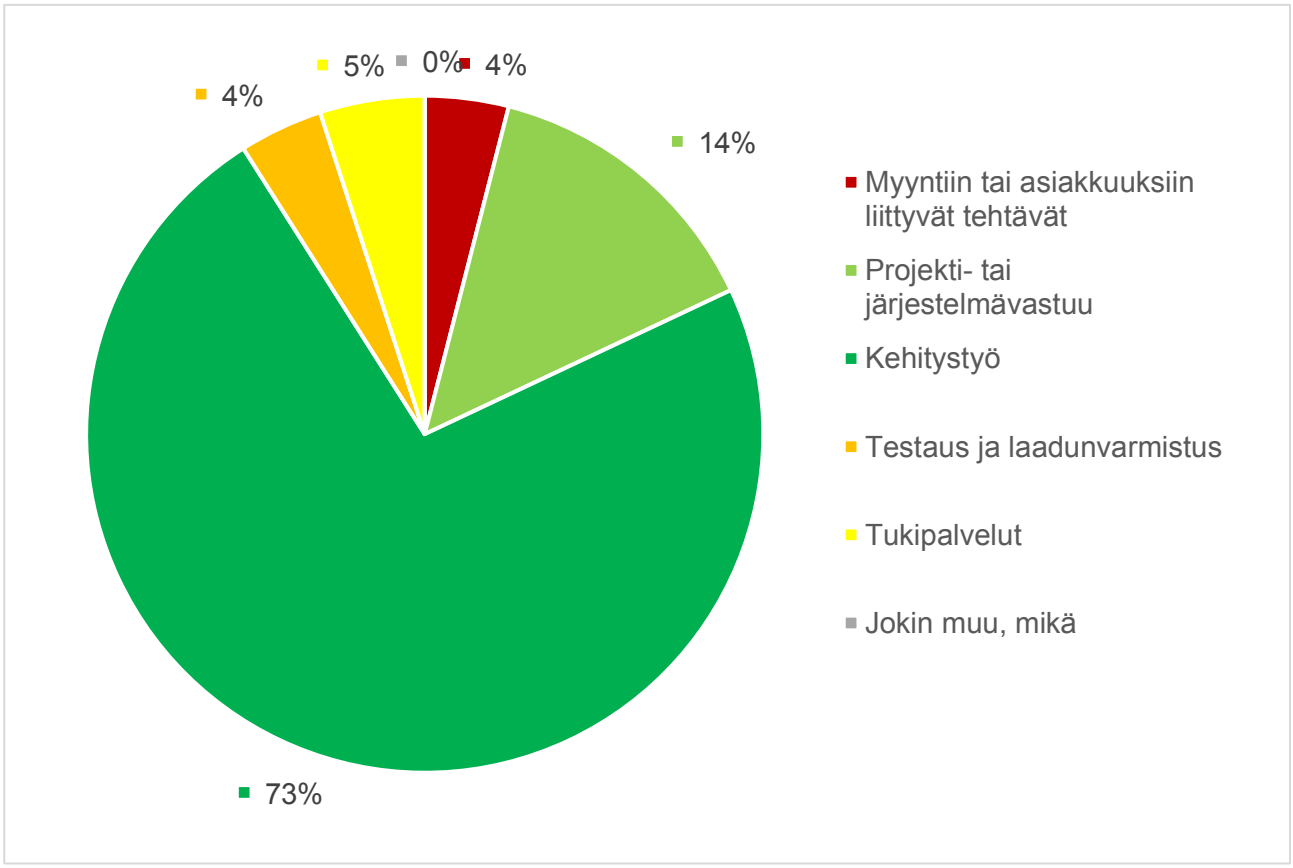
Kyselyyn vastaajien työkokemus ohjelmistoalalta vaihteli alle yhdestä vuodesta yli kahteenkymmeneen vuoteen. Suurimmat ryhmät 22,7 % osuudella olivat 6–10 vuotta ja 11–15 vuotta ohjelmisto alalla työskennellet henkilöt. Yli 20 vuotta työskennelleitä oli 18,2 % vastaajista. Kuvassa 3. näkyy vastaajien työkokemus.



Kuva 3. Kyselyyn vastanneiden työkokemus ohjelmistoalalta

Kyselyyn vastanneista 63,6 % oli toimihenkilöitä ja 22,7 % työntekijöitä. Johtajia oli 9,1 % vastaajista ja yrittäjiä 4,6 %. Toimihenkilöiden osuus oli siis varsin merkittävä vertailtuna muihin ryhmiin.

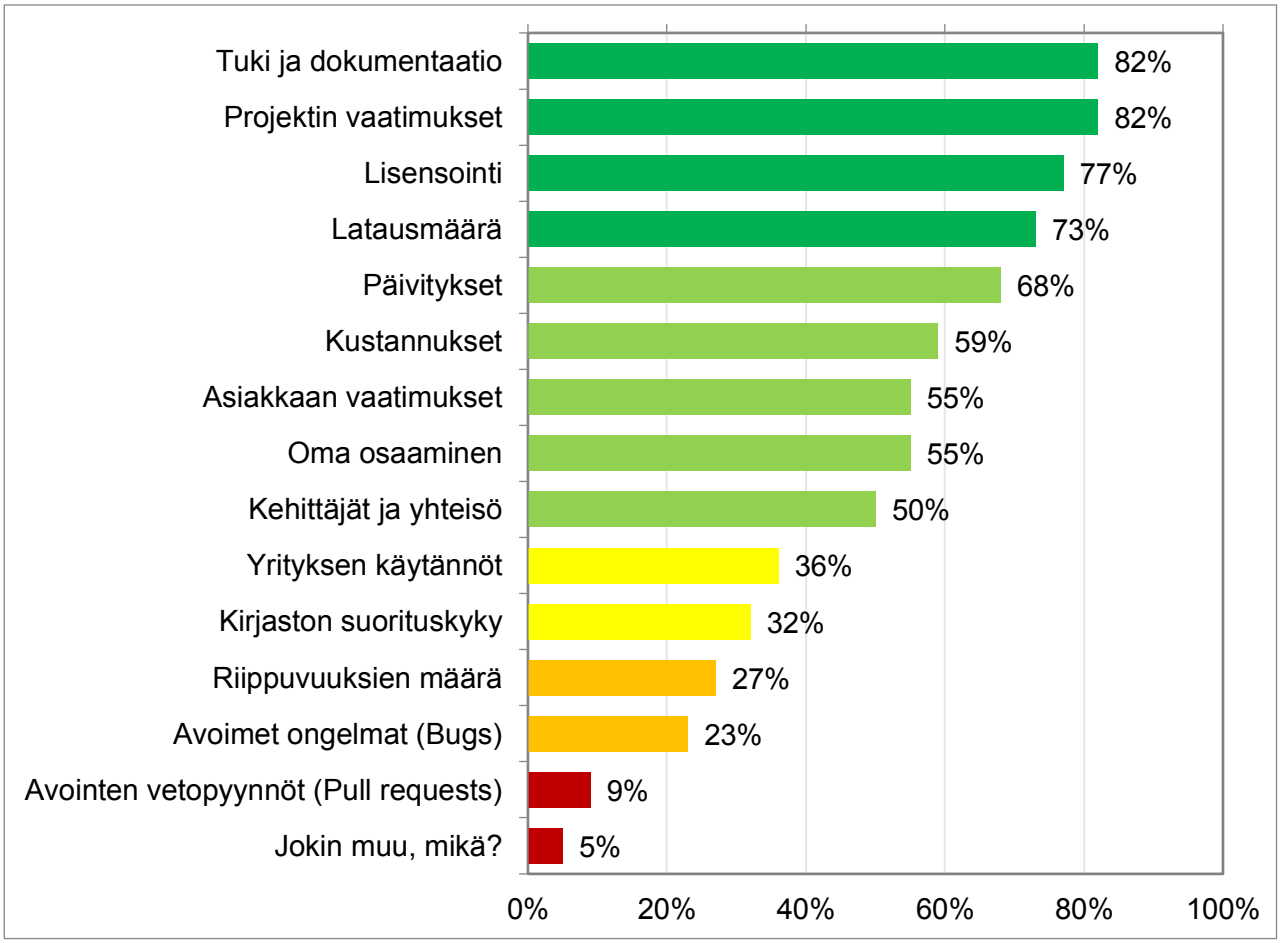
Kuten kuvassa 4. voidaan nähdä, kehitystyö oli pääasiallisena työnkuvana 72,7 % vastaajista. Kehitystyöllä tarkoitetaan tässä yhteydessä esimerkiksi ohjelmistokehittäjiä tai -arkkitehteja sekä ohjelmistokehitystä tekeviä konsultteja. Projekti- tai järjestelmävastuussa oli 13,6 % vastaajista. Projektivastuun työtehtävät ovat esimerkiksi projektipäälliköitä tai tuoteomistajia. Loput vastaajista työskentelivät myynnin, testauksen tai tukipalvelujen parissa. Tukipalveluilla tarkoitetaan esimerkiksi asiakastukea ja ylläpitotöitä.



Kuva 4. Kyselyyn vastanneiden pääasiallinen työnkuva

4.3 Ohjelmistokirjastojen valinta

Kuvassa 5. on havainnollistettu miten vastaajat valitsevat ohjelmistokirjaston. Prosenttiosuus kuvaa sitä, miten suuri osa vastaajista kertoi käyttäneensä kyseessä olevaa valintaperustetta. Ohjelmistokirjastojen valintaan käytetyissä perusteissa oli paljon vaihtelua ja kaikki vaihtoehdot ovat olleet valintaperusteena jollekin vastaajalle. Projektin vaatimukset sekä ohjelmistokirjaston tuki ja dokumentaatio nousivat esille useimpien vastaajien käyttämänä valintaperusteena ja 81,8 % vastaajista kertoi käyttävänsä niitä ohjelmistokirjaston valinnassa. Lisensoinnin osuus oli 77,3 % ja latausmäärä oli puolestaan valintaperuste 72,7 % vastaajista. Ohjelmistokirjaston päivitykset olivat valintaperusteena 68,2 % ja kustannukset 59,1 % vastaajista. 55 % prosentin osuuteen nousivat asiakkaan vaatimukset ja vastaajan oma osaaminen. Puolet vastaajista käytti valintaperusteena ohjelmistokirjaston kehittäjiä ja siihen liittyvää yhteisöä valintaperusteena. Vähemmän käytettyjä valintaperusteita olivat ohjelmistokirjaston riippuvuuksien määrä, avoimet vetopyynnöt (pull requests) ja ongelmat (bugs). Yksi vastaajista käytti valintaperusteena ohjelmistokirjaston koon suhdetta projektin kokoon.

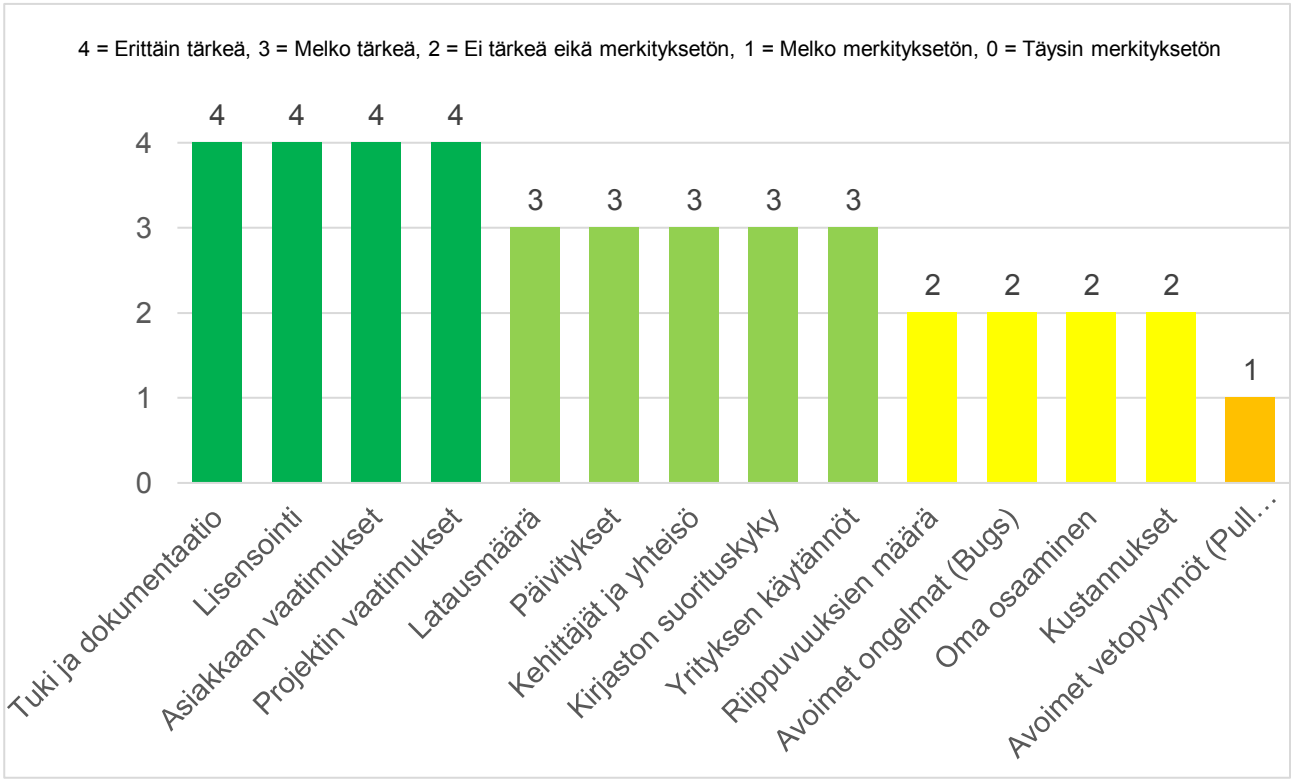


Kuva 5. Kuinka suuri osuus vastaajista käytti tiettyä valintaperustetta

Kun tarkastellaan ohjelmistokirjaston valintaperusteiden käyttöä vastaajien työkokemuksen mukaan, niin erot olivat melko pieniä. Joidenkin valintaperusteiden kohdalla työkokemukselle oli kuitenkin merkitystä. Ohjelmistokirjaston käytöstä aiheutuvat kustannukset olivat tärkeitä kokeneille kehittäjille. Kaikki 16–20 vuotta ja yli 20 vuotta ohjelmistoalalla työskennelleet käyttivät kustannuksia ohjelmistokirjaston valintaperusteena. Myös 3–5 vuoteen alalla työskennelleistä 83 % käytti kustannuksia valintaperusteena. Muiden työkokemusvaihtoehtojen osuudet olivat pienemmät. Lisäksi kaikki yli 20 vuotta alalla työskennelleet ottivat asiakkaan vaatimukset huomioon valinnassa. Tuki ja dokumentaatio oli tärkeä lähes kaikille. Ainoastaan kokemattomat alle kaksi vuotta alalla työskennelleet eivät käyttäneet tukea ja dokumentaatiota niin usein valintaperusteena.

4.4 Valintaperusteiden merkitys

Kuvassa 6. tarkastellaan eri valintaperusteiden merkitystä, joka on kuvattu eri vastausten mediaanina. Näin eniten vastauksia saaneet vaihtoehdot erottuvat selkeästi ja ääripäiden vastauksien osuus ei muodostu merkittäväksi. Jos merkitystä arvioidaan keskiarvon perusteella, niin eri tekijöiden keskinäisessä järjestyksessä on havaittavissa pieniä eroja.



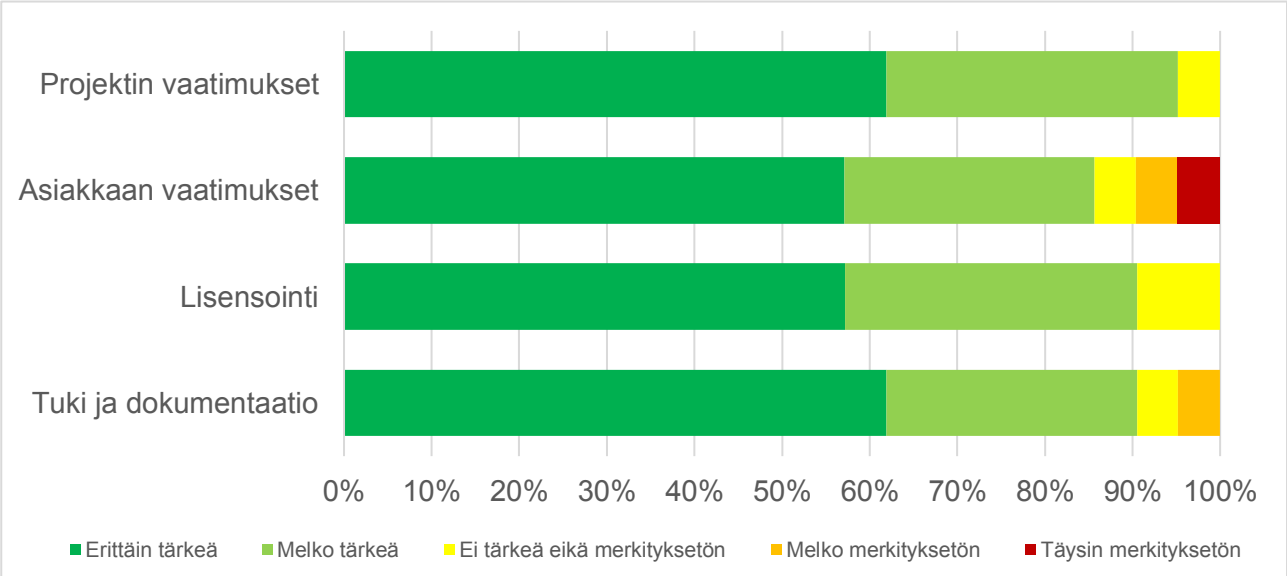
Kuva 6. Valintaperusteiden merkitys

Erittäin tärkeitä valintaperusteita olivat asiakkaan ja projektin asettamat vaatimukset ohjelmistokirjastolle. Ohjelmistokirjaston lisensointi sekä tuki ja dokumentaatio koettiin myös erittäin tärkeiksi valintaperusteiksi. Nämä neljä tekijää nousivat aineistosta merkityksellisimmiksi valintaperusteiksi. On kuitenkin huomionarvoista. että vastausten keskiarvolla mitattuna ero päivityksien ja asiakkaan vaatimusten välillä on melko pieni. Päivityksiä voidaan siis pitää tärkeänä valintaperusteena, vaikka vastauksissa oli enemmän hajontaa vaihtoehtojen kesken.

Melko merkityksellisiä valintaperusteita oli hiukan enemmän. Kuten todettu ohjelmistokirjaston aktiivinen päivittäminen ja julkaistujen versioiden määrä koettiin melko merkitykselliseksi valintaperusteeksi. Valintaan vaikuttivat ohjelmistokirjaston latausmäärä kyseisen ohjelmointikielen paketinhallinasta. Ohjelmistokirjaston ympärillä toimivia kehittäjiä ja yhteisöä pidettiin melko merkityksellisenä samoin kuin ohjelmistokirjaston suorituskykyä. Lisäksi vastaajat katsoivat oman työnantajansa käytännöt melko merkityksellisiksi.

Muilla kuvassa 6. esitetyillä valintaperusteilla oli vähemmän merkitystä. Vastaajien mielestä ohjelmistokirjaston riippuvuuksien ja avoimien ongelmien merkitys ei ollut tärkeä eikä merkityksetön. Myös oma osaaminen ja kustannukset eivät juurikaan vaikuttaneet valintaan. Avoimien vetopyyntöjen määrä GitHubissa koettiin lähes merkityksettömäksi valintaperusteeksi.

Vastaajien mielestä neljää tärkeintä valintaperustetta kannattaa tarkastella vielä lähemmin. Kuvassa 7. huomataan, että vastaukset jakaantuvat lähes kokonaan erittäin tärkeän ja melko tärkeän vaihtoehdon kesken. Esimerkiksi 95,2 % vastaajista piti projektin vaatimuksia vähintään melko tärkeänä valintaperusteena. Lisäksi yli puolet vastaajista on pitänyt näitä neljää valintaperustetta erittäin tärkeinä. Vain hyvin harvat vastaajat kokivat kuvaajassa 5. näkyvät valintaperusteet täysin merkityksettömäksi, merkityksettömäksi tai melko merkityksettömäksi.



Kuva 7. Neljän tärkeimmän valintaperusteen merkitys

Neljässä tärkeimmässä vaihtoehdossa vastauksissa oli hieman vähemmän hajontaa kuin muissa vaihtoehdoissa. Projektin vaatimukset sekä tuki ja dokumentaatio -vastausvaihtoehdoilla variaatiosuhde oli 0,38. Lisensoinnissa ja asiakkaan vaatimuksissa vastaava luku oli 0,43. Vähemmän tärkeissä valintaperusteissa hajontaa oli enemmän ja niiden variaatiosuhde vaihteli 0,52–0,67 välillä. Mitä pienempi variaatiosuhde on, niin sitä vähemmän vastausvaihtoehdoissa on hajontaa. Variaatiosuhde vaihtelee siis nollan ja yhden välillä.

4.5 Näkemyksiä ohjelmistokirjaston valinnasta

Ohjelmistokirjaston valintaa käsitteli yksi avoin kysymys, jonka vastaukset vaihtelivat yhden virkkeen toteamuksesta pitkään yksityiskohtaiseen selvitykseen valintaprosessista. Vastauksissa käsiteltiin suurimmaksi osaksi samoja asioita kuin kyselyssä muutenkin, mutta vastaajat avasivat omia näkemyksiään laajemmin. Vastauksissa tuli ilmi myös uusia asioita, joita monivalintakysymyksissä ei ollut huomioitu. Näitä asioita olivat käyttömukavuus, kirjaston koko suhteessa saavutettuun hyötyyn, testaus, soveltuvuus ongelmanratkaisuun ja yhteensopivuus kehitettävään ohjelmistoon.

Vastaajat pitivät ohjelmistokirjaston latausmäärää merkityksellisenä valintaperusteena, koska suuri latausmäärä kertoo ohjelmistokirjaston laajasta käytöstä. Sen katsottiin ennakoivan ohjelmistokirjastolle pidempää elinkaarta, nopeampaa reagointia ongelmiin ja aktiivisia päivityksiä. Latausmäärien katsottiin vaikuttavan myös muihin valintaperusteisiin: ”Latausmäärät, päivitykset ja kehittäjät/yhteisö kulkevat käsikädessä. Jos kyseessä on suosittu aktiivisesti kehitetty ja ylläpidetty ohjelmistokirjasto siltä löytyvät nämä kaikki ominaisuudet.”

Ohjelmistokirjaston aktiivista päivittämistä pidettiin tärkeänä erityisesti ongelmatilanteissa kuten tietoturvan ja haavoittuvuuksien kohdalla. Ohjelmistokirjastot, joita ei kehitetä, katsottiin uhkaksi tietoturvalle. Eräs vastaaja kertoi valitsevansa kirjaston ensisijaisesti päivitysten ja haavoittuvuuksien korjauksien perusteella. Toisaalta GitHubissa ilmoitettujen avoimien bugien ei katsottu estävän ohjelmistokirjaston käyttöä, jos muut valintaan vaikuttavat asiat ovat kunnossa.

Ohjelmistokirjaston ympärillä olevaan yhteisöä pidettiin merkityksellisenä varsinkin, jos jokin suurempi luotettu taho tukee kirjaston kehitystyötä. Yhteisön katsottiin edesauttavan mahdollisten ongelmien havaitsemisessa ja korjauksessa. Eräs vastaaja kertoi kollegoiden ja tuttujen mielipiteen vaikuttavan ohjelmistokirjaston valintaan. Vastaajan oma yhteisö oli siis merkityksellinen tekijä.

Vastaajat kertoivat, että ohjelmistokirjaston valintaan vaikuttaa se, että kuinka paljon ja millaisia riippuvuuksia ohjelmistokirjaston mukana tulee. Mukana tulevilla riippuvuuksilla katsottiin olevan vaikutusta tietoturvaan, ohjelmiston nopeuteen ja kokoon. Riippuvuuksien kerrottiin kasvattavan ohjelmiston hyökkäyspintaa, vaikka valitussa kirjastossa itsessään ei olisi haavoittuvuuksia.

Lisensointia pidettiin tärkeänä valintaperusteena, joka vaikuttaa mahdollisuuteen käyttää ohjelmistokirjastoa. Lisenssiehtojen nähtiin rajoittavan ohjelmistokirjaston käyttöä, jos ne velvoittavat julkaisemaan lähdekoodin avoimeksi. Vastaajat katsoivat, että kaupallisten ohjelmistokirjastojen käyttäminen tulisi tehdä tarveharkinnan perusteella. Osa vastaajista suosi avoimen lähdekoodin kirjastoja.

Ohjelmistokirjaston dokumentaation tärkeys nousi esille aineistosta. Hyvän dokumentaation katsottiin helpottavan valintaa, kun ohjelmistokirjaston käyttöä voitiin arvioida dokumentaation ja annettujen esimerkkien perusteella. Vastaajat kertoivat, että dokumentaatio vähentää tarvetta tutkia ohjelmistokirjaston lähdekoodia tarkemmin.

Asiakkaan vaatimuksiin ohjelmistokirjaston kohdalla suhtauduttiin eri tavoin. Eräs vastaaja piti asiakkaan mielipidettä merkityksettömänä, sillä asiakkaan ei katsottu olevan ohjelmistokehityksen ammattilainen. Osa vastaajista piti asiakkaan vaatimuksia tärkeinä ja eräs vastaaja kertoi niiden ohjaavan ohjelmistokirjaston valintaa.

4.6 Muita tunnistettuja valintaperusteita

Aineistosta tuli esille, että käyttömukavuus on merkittävä tekijä ohjelmistokirjaston valinnassa. Ohjelmistokehittäjän näkökulmasta ohjelmistokirjaston käytön piti olla miellyttävää ja sopia ohjelmiston toteutustapaan. Erään vastaajan käyttämä termi ”kehittäjämukavuus”, joka kuvaa ilmiötä erinomaisen hyvin.

Vastaajat katsoivat, että ohjelmistokirjaston koon pitää olla sopivassa suhteessa kehittävän ohjelmiston kokoon. Samalla tavalla tärkeänä pidettiin soveltuvuutta ongelman ratkaisuun. Yksinkertaisen ongelman ratkaisuun ei haluttu käyttää liian suurta kirjastoa. Parempana vaihtoehtona koettiin ratkaista ongelma itse tai käyttää spesifimpää ohjelmistokirjastoa. Ohjelmistokirjaston mukana tulevien riippuvuuksien määrä otettiin huomioon kirjaston koon arvioinnissa.

Yhteensopivuus tuli esille uutena teemana, jota ei ollut huomioitu kyselyn monivalintaosiossa. Vastauksissa puhutaan ohjelmistokirjaston yhteensopivuudesta muiden käytettyjen ohjelmistokirjastojen kanssa. Lisäksi eräs vastaaja puhuu ”ohjelmistoekosysteemin mukaisesta lähestymistavasta”. Tämä voitaisiin käsittää esimerkiksi yhteensopivuutena

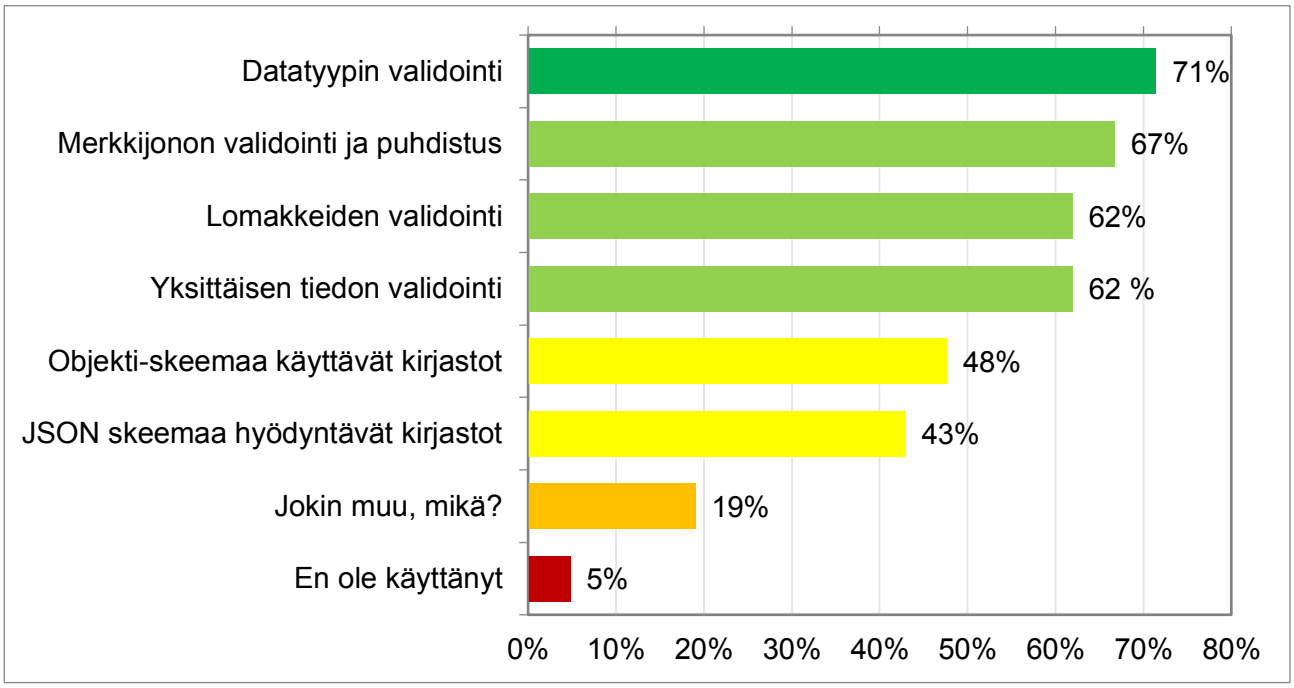
ohjelmistoarkkitehtuuriin ja sovelluksen rajapintoihin. Ohjelmistokirjastojen valintaperusteiden kohdalla laveampi tulkinta on kuitenkin hyödyllisempi. Näin valintaperusteen voidaan käsittää sisältävän kaikki ohjelmistokirjaston yhteensopivuuteen liittyvät asiat.

Lisäksi testauksen katsottiin vaikuttavan ohjelmistokirjaston valintaan. Testauksesta puhutettiin ohjelmistotestauksena ja ohjelmistokirjaston käytöstä syntyneenä kokemuksena. Varsinainen ohjelmistotestaus eli ohjelmistokirjastolle kirjoitutetut testit mainittiin valintaan vaikuttavana seikkana. Niiden lisäksi huomioitiin ohjelmistokirjaston käyttö ja sitä kautta tuleva käytännön testaus. Kun ohjelmistokirjastoa käytetään paljon, niin mahdollisuus virheiden löytymiseen kasvaa. Näin ollen kirjastoa voidaan käyttää vastaajien mielestä huolettomammin.

4.7 Validointiin käytettävän ohjelmistokirjaston valinta

4.7.1 Validointikirjastojen käyttö

Kuvasta 8. voidaan todeta, että vastaajat olivat käyttäneet laajasti erityyppisiä validointiin käytettäviä kirjastoja. Datatyypin validointiin käytettäviä kirjastot olivat yleisimmin käytettyjä ja 71,4 % vastaajista oli käyttänyt niitä. Käytännön esimerkki datatyypin validoinnista on se, että numeroille tarkoitettuun kenttään voi syöttää vain numeroita eikä tekstiä tai muita merkkejä. Seuraavaksi suosituimpia olivat merkkijonon validointiin käytettävät kirjastot 66,7 % osuudella. Molempia lomakkeiden validointiin ja yksittäisen tiedon validointiin käytettäviä kirjastoja oli hyödyntänyt 61,9 % kyselyyn vastanneista. Lomakkeiden validointiin käytettävistä kirjastoista voisi olla esimerkki suosittu JavaScript-kirjasto React-Hook-Forms. Yksitäisen tiedon validoinnilla tarkoitetaan vaikkapa sähköpostiosoitteen ja sosiaaliturvatunnuksen oikeellisuuden tarkistusta.



Kuva 8. Validointikirjastojen käyttö vastaajien keskuudessa

Objekti-skeemaan perustuvia kirjastoja oli käyttänyt alle puolet vastaajista. Objekti-skeemaa hyödyntäviä kirjastoja oli käyttänyt 47,6 % vastaajista. Hieman vähemmän 42,9 % oli hyödyntänyt JSON skeemaan perustuvia validointikirjastoja. 1–2 vuotta ohjelmistoalalla olleista vastaajista kaikki olivat käyttäneet JSON skeemaan perustuvia ohjelmistokirjastoja. 11–15 vuotta alalla olleiden kohdalla vastaava luku oli 80 %. Muiden työkokemusvuosien kohdalla JSON-skeeman käyttö oli vähäisempää. Vaikka skeemaan perustuvia ohjelmistokirjastoja käytettiin melko laajasti, niin niiden käyttö oli merkittävästi vähäisempää kuin datatyypin validointiin käytettävien kirjastojen.

Aineistosta nousi esiin validointiin käytettäviä ratkaisuja, joita ei ollut huomioitu valmiissa vastausvaihtoehdoissa. Joitakin muita kuin vastausvaihtoehtoissa tarjottuja validointikirjastotyyppejä oli käyttänyt 19 % vastaajista. Huomionarvoisena voidaan nostaa ohjelmistokehyksien tarjoavat validointivaihtoehdot kuten esimerkiksi Spring Bootin Java Bean Validation. Kuvasta 8. huomataan, että vain 4,8 % vastaajista ei ollut käyttänyt validointiin mitään ohjelmistokirjastoa.

4.8 Valinnassa huomioituja asioita

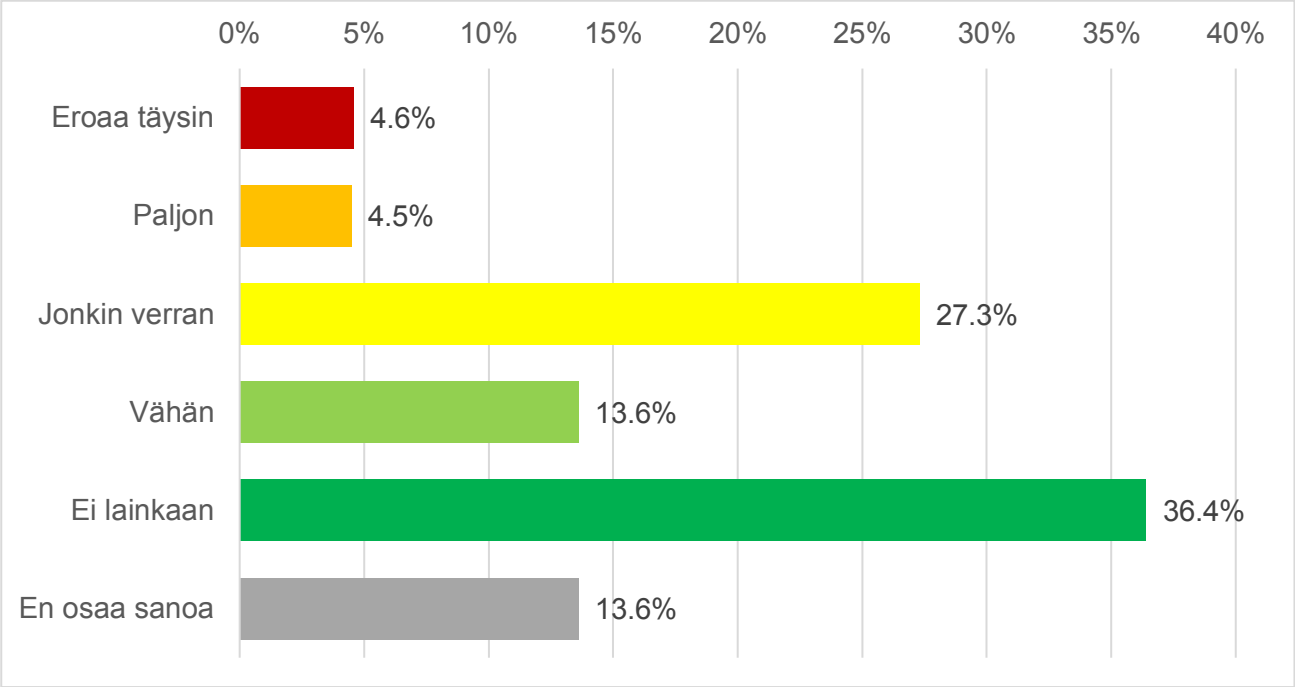
Validointiin käytettävän ohjelmistokirjaston valintaa käsittelevässä avoimessa kysymyksessä vastaajat kertoivat kiinnittävänsä huomiota hyvin erilaisiin asioihin. Ohjelmistokehyksen tarjoamat validointimahdollisuudet nousivat esiin tässäkin kysymyksessä. Muita aiemmin tunnistettuja teemoja olivat asiakkaan vaatimukset, käyttömukavuus ja yhteensopivuus. Ohjelmistokehyksessä

mukana olevien validointimahdollisuuksien käyttöä perusteltiin sillä, että silloin pystyttiin välttämään ylimääräisiä ohjelmistokirjastoja. Eräs vastaaja kertoi tietyn validointikirjaston olevan asiakkaan vaatimus. Käyttömukavuus toistui useammassa vastauksessa ja vastaajat nostivat esille erityisesti käyttöönoton ja käytön helppouden. Yhteensopivuudessa vastaajat nostivat esille TypeScript tuen ja yhteensopivuuden muiden ohjelmistokirjastojen kanssa.

Ainoa huomionarvoinen täysin uusi teema oli varmuus validoinnin toimivuudesta. Vastaajat halusivat olla varmoja siitä, että valittu ohjelmistokirjaston suorittaa validoinnin varmasti oikein ja hyvin. Aineistosta ei kuitenkaan selvinnyt mitä tällä tarkkaan ottaen tarkoitettiin. Eräs vastaaja kertoi, että arvioi validointiin käytettäviä ohjelmistokirjastoja samoin perustein kuin muitakin ohjelmistokirjastoja.

4.8.1 Erovaisuudet valinnassa

Kuten kuvasta 9. voidaan tulkita 36,4 % vastaajista katsoi, että validointiin käytettävän kirjaston valinta ei eroa lainkaan muiden ohjelmistokirjastojen valinnasta. 13,6 % vastaajista mielestä valinta erosi vain vähän. Yhteensä siis 50 % vastaajista oli siis sitä mieltä, että valintaperusteet eroavat vähän tai ei lainkaan. Alla viisi prosenttia katsoi, että valinta eroaa täysin muiden ohjelmistokirjastojen valinnasta. Samoin vain 4,6 % vastaajista katsoi, että eroavaisuuksia oli paljon. Välimaastoon asettautui lähes kolmannes eli 27,3 % vastaajista, joiden mielestä on jonkin verran eroavaisuuksia. Yli kymmenen prosenttia vastaajista ei osannut päättää kantaansa.



Kuva 9. Kuinka paljon validointiin käytettävän kirjaston valinta eroaa muiden ohjelmistokirjastojen valinnasta

Validointiin käytettävän ohjelmistokirjaston ja muuhun tarkoitukseen käytettävän ohjelmistokirjaston valinnan eroja käsitteli yksi avoin kysymys. Kun edellisessä kysymyksessä tarkasteltiin, miten paljon validointiin käytettävän ohjelmistokirjaston valinta eroaa muista ohjelmistokirjastoista, niin tämän kysymyksen tarkoituksena oli saada syventävää tietoa eroavaisuuksista. Valitettavan usein aineistossa ei käsitelty asioita, jotka olisivat koskeneet juuri validointikirjastojen valinnan eroja, vaan vastauksissa puhuttiin ohjelmistokirjastojen valinnasta yleisemmällä tasolla.

Ohjelmistokehyksien tarjoamien validointiratkaisujen käyttö nousi uudestaan esiin, vaikka se ei varsinaisesti olekaan erovaisuus validointiin käytettävien ohjelmistokirjastojen valinnassa. Avoimien ongelmien määrä koettiin validointikirjaston kohdalla tärkeämpänä kuin muiden ohjelmistokirjastojen kohdalla. Vastaajat katsoivat, että verrattuna muihin ohjelmistokirjastoihin validointikirjaston tuli olla skaalautuva ja mukautettavissa muuttuviin käyttötarkoituksiin. Soveltuvuus käyttötarkoitukseen ja suorituskyky koettiin tärkeiksi ominaisuuksiksi juuri validointiin käytettävän ohjelmistokirjaston valinnassa. Näiden lisäksi validointi nähtiin olennaisena osana ohjelmakoodin muodostumisessa ja se erosi siten muiden ohjelmistokirjastojen valinnasta. Aktiivinen kehitys, päivitykset ja erityisesti tietoturvapäivitykset mainittiin myös tässä yhteydessä, mutta niiden katsottiin koskevan kaikkien ohjelmistokirjastojen valintaa.

4.9 REST-rajapinnat ja validointikirjaston valinta

Kysymykseen REST-rajapinta-arkkitehtuurin vaikutuksesta validointikirjaston valintaan vastaajat saivat muotoilla vastauksensa vapaamuotoisesti. Vastaajien näkemykset poikkesivat hyvin paljon toisistaan, mutta toisaalta joissakin asioissa oltiin samaa mieltä. Hyvä esimerkki on se, että eräs vastaaja katsoi, että REST-arkkitehtuuri ei vaikuttanut millään tavalla datastruktuuriin ja siksi sillä ei ollut merkitystä validointikirjaston valinnassa. Osa vastaajista taas luetteli useampia asioita, joissa REST-rajapinnoilla oli vaikutusta validointikirjaston valintaan.

Useissa vastauksissa puhuttiin käyttömukavuudesta, johon aiemmin viitattiin myös termillä kehittäjämukavuus. Käyttömukavuudeksi laskettiin ohjelmakoodin ylläpidettävyys, ymmärrettävyys, validointikirjaston käytön helppous ja oman työn helpottaminen. Erän vastaajan mielestä omaa työtä helpotti määrittelyntyön väheneminen, kun asiat oli otettu huomioon validointikirjastossa. Määrittelytyön väheneminen säästi myös aikaa. Vastaajat katsoivat siis, että REST-rajapinnat vaikuttavat validointikirjaston valintaan ainakin käyttömukavuuden osalta.

Osa vastaajista piti tietoturvaa merkittävänä valintaperusteena, kun validointikirjaston valintaa tarkasteltiin REST-rajapintojen näkökulmasta. Tietoturvaa pidettiin ehdottomana vaatimuksena validointikirjastolle. Vasta hyvän tietoturvan jälkeen oli tarpeellista tarkastella muita valintaperusteita.

Suorituskyky ja skaalautuvuus mainittiin usein tietoturvan yhteydessä. Eräs vastaaja kertoi asiasta seuraavasti: "Nostaisin suorituskyvyn, skaalautuvuuden ja tietoturvan. Jos noista joku pettää, kirjasto ei täytä tehtäväänsä.” Toisaalta suorituskyky ja skaalautuvuus nähtiin asioida, joita piti tarkastella tarveharkinnan mukaan. Skaalautuvuus ja suorituskyky nousivat kuitenkin esille aineistosta asioina, jotka otettiin huomioon REST-rajapintojen ja validointikirjastojen yhteydessä.

Eräs vastaaja kertoi valitsevansa aina validointiin ohjelmistokirjaston, jotka tuottavat tietotyypit ja validoinnin. Hän katsoi sen helpottavan jatkokäyttöä, joka taas liittyy selkeästi skaalautuvuuteen. Tietotyypin ja validoinnin yhteys nousi esille myös toisessa yhteydessä.

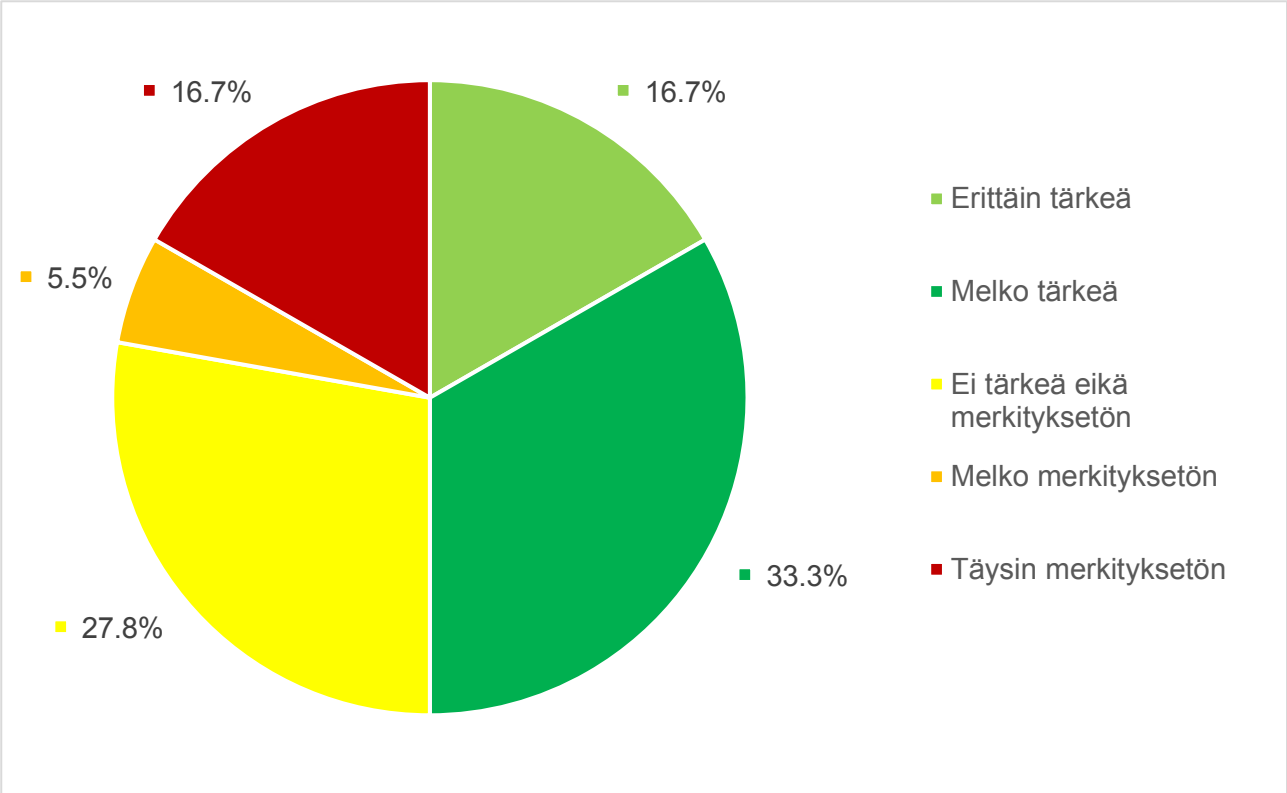
Yksi aineistosta esille noussut teema oli koodin virheettömyys. Eräs vastaaja painotti, että vahvasti tyypitetyissä ohjelmointikielissä validointi ei ole samanlainen kipupiste kuin esimerkiksi JavaScriptissä. JavaScriptiin pohjautuvan TypeScriptin kohdalla tyyppimäärityksen ja validoinnin nähtiin parantavan koodin virheettömyyttä. Vastauksissa suhtauduttiin kuitenkin ristiriitaisesti virheettömyyteen. Osa piti virheettömyyttä ehdottomana, kun taas osan mielestä validoinnissa oli aina bugeja. Aineistosta ei voinut selkeästi tulkita miten virheettömyys vaikutti validointikirjaston valintaan REST-rajapinta-arkkitehtuurissa.

REST-rajapinta-arkkitehtuurin osalta vastaajat ottivat myös huomion validointikirjaston tuottamiin virheviesteihin. Niiden haluttiin olevan asianmukaisia, selkeitä ja helposti muokattavia. Samassa yhteydessä nousi esiin tuki monikielisyydelle. Aikaisemmissa kysymyksissä käsitelty yhteensopivuus muiden ohjelmistokirjastojen kanssa mainittiin myös REST-rajapintojen yhteydessä.

4.9.1 Merkitys validointikirjastojen valinnassa

Kuva 10. havainnollistaa REST-rajapinta-arkkitehtuurin merkitystä validointikirjaston valinnassa vastaajien mielestä. Siitä voidaan havaita, että yhteensä puolet pitää REST-rajapintoja merkityksellisenä tekijänä. 33,3 % katsoo niiden olevan melko tärkeä tekijä valinnassa ja 16,7 % pitää erittäin tärkeänä tekijänä. Vastaavasti kuitenkin saman verran 16,7 % katsoo REST arkkitehtuurin olevan täysin merkityksetön tekijä validointikirjaston valinnassa. Yhtä suuri osuus vastaajista on siis täysin päinvastaista mieltä REST-arkkitehtuurin merkityksestä. 5,5 % näkee

asian melko merkityksettömänä. Lisäksi ääripäiden välille mahtuu vielä 27,8 % prosentin joukko vastaajia, joiden mielestä asia ei ole tärkeä eikä merkityksetön.



Kuva 10. REST-rajapinta-arkkitehtuurin vaikutus validointikirjaton valintaan

Vastauksissa on siis melko paljon hajontaa ja variaatiosuhde onkin 0,67. Tästä huolimatta voidaan sanoa, että hieman suurempi osa vastaajista pitää asiaa tärkeänä kuin merkityksettömänä. Kokeneet 16–20 vuotta ja yli 20 vuotta ohjelmistoalalla työskennelleet ihmiset pitivät REST-rajapinta-arkkitehtuuria hieman tärkeämpänä kuin muut vastaajat. Heidän vastauksensa osuivat mediaanilla vaihtoehdolle erittäin tärkeä.

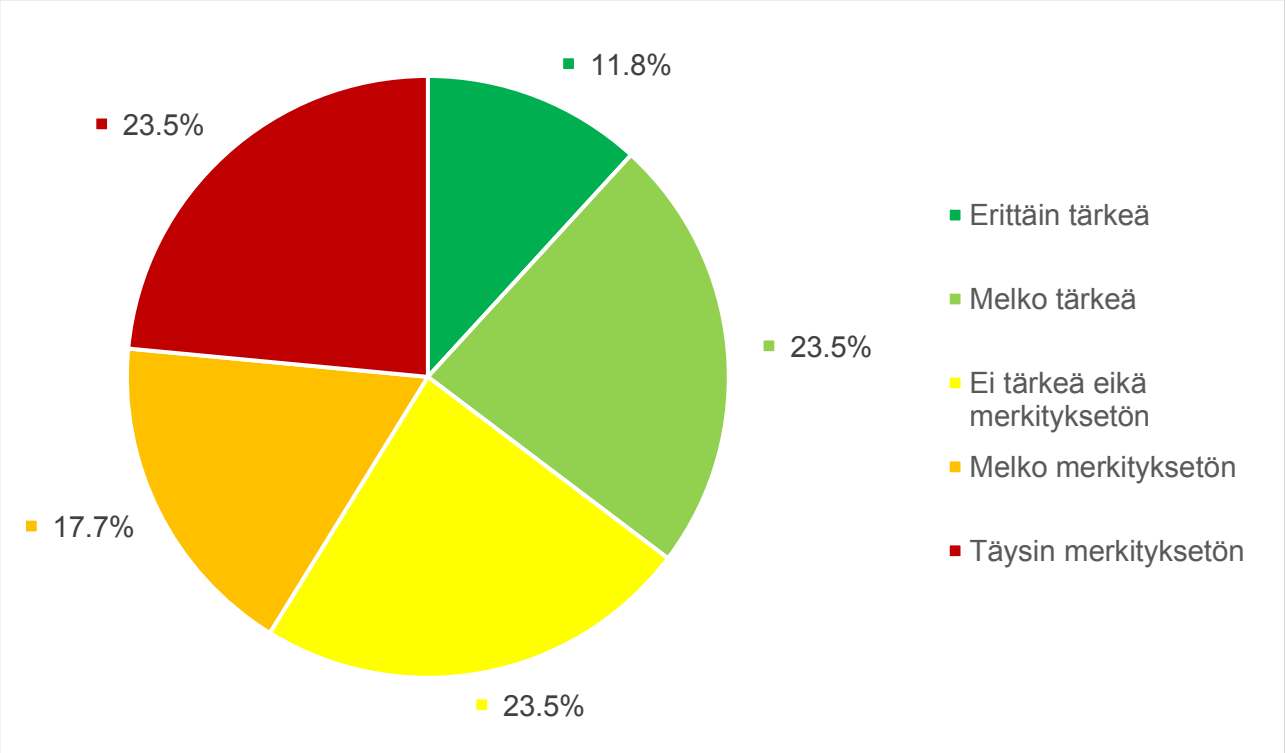
4.10 JSON-datamuoto ja validointikirjastot

Aineistosta oli tulkittavissa, että vastaajat eivät pitäneet JSON-datamuotoa kovin merkityksellisenä tekijänä validointikirjaston valinnassa. Datamuotoa ei pidetty ylipäätään kovin merkityksellisenä asiana, jos käytettiin erillistä validointikirjastoa. Osa vastaajista jopa katsoi, että tiedonsiirtomuodon ei tule vaikuttaa millään tavalla validointiin. Validointi haluttiin toteuttaa niin, että datamuodolla ei olisi merkitystä toteutuksessa. Vastaajat kertoivat myös heidän käyttämiensä ohjelmistokirjastojen tukevan JSON-datamuotoa. Ainoastaan yksi vastaaja pohti, että erittäin suurten JSON

datamäärien kohdalla ohjelmistokirjaston suorituskyky tulisi ottaa huomioon. JSON datamuotoa ei siis juurikaan huomioitu validointikirjaston valinnassa.

4.11 Merkitys validointikirjaston valinnassa

JSON datamuodon merkityksestä oltiin erimielisiä vastaajien keskuudessa. Mielipiteet jakaantuivat melko tasaisesti kaikkien vaihtoehtojen välille, joka voidaan havaita myös kuvasta 11. Vastauksissa oli huomattavan paljon hajontaa ja variaatiosuhde oli korkea 0,76.



Kuva 11. JSON datamuodon merkitys validointikirjaston valinnassa

Kuvan 11. mukaan vaihtoehdot melko tärkeä, ei tärkeä eikä merkityksetön ja täysin merkityksetön olivat tasoissa 23,5 % osuudella. 17,7 % vastaajista piti JSON datamuotoa melko tärkeänä tekijänä validointikirjaston valinnassa. Erittäin tärkeänä asiaa piti vain 11,8 % vastaajista. Hieman yksinkertaistaen voidaan sanoa, että vastaajat eivät pitäneet asiaa kovin merkityksellisenä, johon myös vastausten mediaani asettuu. Melko merkityksetön ja täysin merkityksetön saivat yhdessä 41,2 % osuuden, joka on hieman suurempi kuin erittäin tärkeän ja melko tärkeän yhteenlaskettu 35,3 % osuus. Lisäksi on huomattava, että ei tärkeänä eikä merkityksettömänä asiaa piti 23,5 % vastaajista. JSON-datamuodon kohdalla vastaajien työkokemuksella ei ollut juuri merkitystä.

5 Pohdinta

Tutkimuksen tavoitteena oli saada selkeä kuva ohjelmistoalalla työskentelevien henkilöiden käyttämistä valintaperusteista ohjelmistokirjastojen valinnassa. Tutkimuskysymyksiin oli tarkoitus saada selkeitä vastauksia ohjelmistoalalla työskentelevien ihmisten mielipiteiden ja aikaisempien tutkimusten perusteella. Osaan tutkimuskysymyksistä saatiin uutta ja syventävää tietoa. Joihinkin tutkimuskysymyksiin ei pystytty vastamaan kovin syvällisesti ja yksityiskohtaisesti. Kaikkiin tutkimuskysymyksiin saatiin kuitenkin vastauksia, mutta niiden taso vaihteli jonkin verran.

5.1 Yhteenveto tutkimuskysymyksien perusteella

Tutkimusaineiston valossa validointiin käytettävien ohjelmistokirjastojen valinta ei eroa kovinkaan paljon muiden ohjelmistokirjastojen valinnasta. Valintaa ohjaavat suuret linjat ovat samanlaisia molemmissa tapauksissa. Tutkimuksessa tunnistetut neljä yleisimmin käytettyä ohjelmistokirjaston valintaperustetta ohjaavat hyvin myös validointikirjaston valintaa. Projektin ja asiakkaan vaatimukset ovat hyvä ohjenuora minkä tahansa ohjelmistokirjaston valinnalle. Lisensointi määrittää valintaa riippumatta ohjelmistokirjaston käyttöpatauksesta. Hyvästä tuesta ja dokumentaatiosta on hyötyä kaikkien ohjelmistokirjastojen käytössä.

Neljän tärkeimmän valintaperusteen lisäksi muut yleisesti käytetyt ohjelmistokirjaston valintaperusteet soveltuvat hyvin validointiin käytettävien ohjelmistokirjastojen kohdalla. Ainakin ohjelmistokirjaston latausmäärä, aktiiviset päivitykset ja kehittäjäyhteisö ovat asioita, jotka kannattaa ottaa huomioon validointikirjaston ja ylipäätänsä minkä tahansa ohjelmistokirjaston valinnassa. Neljän tärkeimmän valintaperusteen lisäksi edellä mainitut asiat nousevat selvästi esiin, kun tarkastellaan ohjelmistokirjastojen käyttöä ja eri valintaperusteiden merkitystä.

Vaikka valintaperusteissa ei ole suuria eroja, niin aineiston perusteella voitiin tunnistaa seikkoja, jotka painottuvat juuri validointiin köytettävien ohjelmistokirjastojen kohdalla. Edellisessä kappaleessa mainitut päivitykset ovat tärkeitä validointikirjaston valinnassa ennen kaikkea tietoturvan näkökulmasta. Samaan asiaan sivuaa myös se, että validoinnin oikeellisuudesta halutaan olla varmoja. Lisäksi validointikirjaston kohdalla on tärkeää huomioida suorituskyky, skaalautuvuus ja mukautuvuus erilaisiin käyttötarkoituksiin. Lisäksi tutkimusaineistosta esiin nousee esiin se, että monissa tapauksissa voidaan käyttää sovelluskehyksen tarjoamia validointivaihtoehtoja. Silloin validointiin ei tarvita erillistä ohjelmistokirjastoa.

Puolet kyselytutkimukseen vastanneista oli sitä mieltä, että REST-rajapinta-arkkitehtuuri tulisi ottaa huomioon validointikirjastoa valittaessa. Tutkimusaineiston perusteella voidaan sanoa, että REST-rajapintojen kohdalla tietoturva, skaalautuvuus ja suorituskyky ovat tärkeitä näkökohtia

validointikirjaston valinnassa. Validointikirjaston käytön ja koodipohjan ylläpitämisen helppous oli vastaajille oleellista ja siitä puhuttiin käyttömukavuutena tai kehittäjämukavuutena. Vastaajien mielestä REST-rajapinnoilla validointikirjaston kirjaston tulisi tuottaa tietotyypit ja validoinnin. Näin voidaan helpottaa jatkokäyttöä.

JSON-datamuodon merkitys validointikirjaston valinnassa on melko pieni. Aineiston perusteella siitä on vaikea tehdä selkeitä yleistyksiä, sillä vastauksissa oli niin paljon hajontaa. Voidaan kuitenkin sanoa, että JSON-datamuodon ei katsottu vaikuttavan validoitiin ja sen ei haluttu vaikuttava toteutettavaan validointiratkaisuun.

5.2 Tuloksiin perustuvat johtopäätökset

Kyselytutkimuksen tulosten perusteella voidaan todeta, että ohjelmistokirjastojen valintaa ohjaavat monet eri tekijät. Kuten sanottu useimmat niistä pätevät aivan yhtä hyvin validointikirjaston valintaan kuin jonkin muun ohjelmistokirjaston valintaan. Erot näiden kahden välillä ovat vähäisiä ja tuntuvat perustuvan ohjelmistoalan ammattilaisten mieltymyksiin painottaa valinnassa eri asioita. Yhtään kuvassa 5. tarkasteltua ohjelmistokirjaston valintaperustetta ei voida sulkea pois validointikirjastojen kohdalla. Lisäksi kyselytutkimuksen avoimissa kysymyksissä esiin tulleita valintaan vaikuttavia tekijöitä käsiteltiin monessa yhteydessä niin validointikirjastojen kuin muidenkin ohjelmistokirjastojen kohdalla. Esimerkiksi päivityksiä, haavoittuvuuksia ja tietoturvaa käsiteltiin yleisesti ohjelmistokirjastojen yhteydessä sekä validointikirjastojen ja REST-rajapintojen kohdalla.

Validointikirjastojen kohdalla valintaperusteita painotetaan hieman eri tavalla kuin muiden ohjelmistokirjastojen kohdalla. Yleisten ohjelmistokirjaston valintaperusteiden lisäksi esimerkiksi yhteensopivuus, käyttömukavuus ja yhteensopivuus olivat teemoja, jotka mainittiin vaikuttavan validointiin käytettävien ohjelmistokirjastojen valintaan. Tulosten perusteella yhteenvetona voidaankin sanoa, että validointikirjastojen valinnassa perusteet ovat samat kuin muidenkin ohjelmistokirjastojen kohdalla. Asiassa tulee kuitenkin ottaa huomioon erityisesti yhteensopivuus, helppokäyttöisyys, skaalautuvuus ja tietoturva. Lisäksi on hyvä käyttää ohjelmistokehyksen tarjoamia mahdollisuuksia validointiin, jos se sopii käyttötapaukseen.

Tulosten perusteella REST-rajapinta-arkkitehtuurilla on merkitystä monessakin asiassa validointikirjaston valinnassa. Päätelmänä on, että tietoturva, suorituskyky ja skaalautuvuus ovat hyvin merkittäviä tekijöitä validointikirjaston valinnassa REST-rajapinnoilla. REST-arkkitehtuuria varten valittavan validointikirjaston tulisi olla helppokäyttöinen, ymmärrettävä ja sen tulisi tuottaa samalla tietotyypit ja validointi.

JSON-datamuodon merkityksestä validointikirjaston valintaan ei voida tehdä yleistäviä johtopäätöksiä kyselytutkimuksen perusteella. Vastauksissa oli niin paljon hajontaa. Myös avoimiin kysymyksiin vastaukset eivät tuoneet juurikaan lisää tietoa valinnan perusteeksi.

Muutenkin edellisissä kappaleissa esitetyt päätelmät ovat melko karkeita yksinkertaistuksia ja yleistyksiä kyselytutkimuksen tuloksista. Tutkimuksen tulokset antavat kuitenkin reilusti tietoa ohjelmistokirjaston ja validointiin käytettävän ohjelmistokirjaston valinnan perusteeksi. Näin lukijan omille tulkinnoille jää enemmän tilaa ja hän voi hyödyntää myös omaa kokemustaan tulosten tulkinnassa.

5.3 Tulosten peilaaminen tietoperustaan

Tutkimuksen tuloksia voidaan vertailla parhaiten Vargasin ja kumppanien tekemään tutkimukseen ohjelmistokirjastojen valinnasta. Tutkimuksessa oli haastateltu kuuttatoista ohjelmistokehittäjää. Tähän tutkimuksen vastaajista valtaosa oli ohjelmistokehittäjiä. Molempien tutkimusten vastaajat ovat siis ammatillisesti melko lähellä toisiaan. Myös Vargasin ja kumppanien tutkimuksen vastaavien kokemusvuodet ovat saman suuntaiset. (2020)

Vargasin ja kumppanien tutkimuksessa valintaperusteita oli jaoteltu hieman eri tavalla kuin tässä tutkimuksessa, mutta yhtäläisyyksiä tuloksissa on helppo löytää. Tuloksissa on myös joitakin hyvin silmiinpistäviä erovaisuuksia. Yhtäläisyyksiä on huomattavasti helpompi ymmärtää, mutta joidenkin erovaisuuksien kohdalla on hankalampi ymmärtää syitä tuloksen erilaisuudessa. Luultavasti kysymyksen asettelussa on ollut merkittäviä eroja.

Tässä tutkimuksessa tunnistetut neljä merkityksellisintä valintaperustetta olivat projektin vaatimukset, asiakkaan vaatimukset, lisensointi sekä tuki ja dokumentaatio. Nämä kaikki neljä löytyvät myös Vargasin ja kumppanien tutkimuksesta. Lisäksi näistä neljästä asiasta voidaan nimetä esimerkit eroavaisuuksista ja yhtäläisyyksistä.

Yhtäläisyyksistä selkeimpiä ovat dokumentaatio ja lisensointi. Vargasin ja kumppanien tutkimuksessa 55 % prosenttia vastaajista oli luokitellut dokumentaation korkean vaikutuksen tekijäksi (2020). Tämä on linjassa tämän tutkimuksen havaintojen kanssa, sillä 61,9 % vastaajista piti tukea ja dokumentaatiota erittäin tärkeänä valintaperusteena. Lisensointi on taloudellisista tekijöistä Vargasin ja kumppanien tutkimuksessa merkittävin tekijä ja 45 % vastaajista katsoi sen olevan korkea vaikutuksen valintaperuste (2020). Tässä tutkimuksessa 57,2 % vastaajista arvotti lisensoinnin erittäin tärkeäksi valintaperusteeksi. Näissä tutkimusten tulokset ovat hyvin samansuuntaiset.

Erovaisuuksista silmiinpistävin on asiakkaiden merkitys. Vargasin ja kumppanien tutkimuksessa vastaajista 7 % ajatteli asiakkaiden olevan korkean vaikutuksen valintaperuste ohjelmistokirjaston valinnassa (2020). Tässä tutkimuksessa asiakkaan vaatimukset olivat erittäin tärkeä valintaperuste 57,1 % vastaajista. Tässä kohtaa ero on hyvin merkittävä. Suurin syy erovaisuuksiin on ehkäpä käytetyissä termeissä. Vargasin ja kumppanien tutkimuksessa kysymyksen asettelussa puhutaan vain asiakkaista, kun tässä tutkimuksessa on käytetty termiä asiakkaan vaatimukset. Tämä on jo sinällään arvolatautunut, kun puhutaan vaatimuksista. Vargasin ja kumppanien tutkimuksessa ei puhuta projektin vaatimuksista laisinkaan. Lähimpänä projektin vaatimuksia voidaan ajatella olevan soveltuvuus tarkoitukseen. Nämä ovat kuitenkin hieman eri asia, joten niiden vertailussa ei ole mieltä.

Samansuuntaisia tuloksia Vargasin ja kumppanien tutkimuksen tuloksissa on muun muassa ohjelmistokirjaston aktiivisissa päivityksissä, yhteisön roolissa sekä siinä miten suosittu ohjelmistokirjasto on. Yhtäläisyyksiä voidaan löytää Vargasin ja kumppanien tutkimuksen haastatteluista nostetuista lainauksista ja tämän tutkimuksen avoimien kysymysten vastauksista. Vargasin ja kumppanien tutkimuksessa eräs vastaaja kertoo, että hänen edustamassa organisaatiossa tietoturvalle on erittäin korkeat standardit ohjelmistokirjaston valinnassa (2020). Tämä on yhtenevää tämän tutkimuksen kanssa, sillä eräs vastaaja kertoi virheettömyyden ja tietoturvan olevan ehdotonta. Lisäksi toinen vastaaja kertoi, että validointikirjasto ei täytä tehtäväänsä, jos tietoturva pettää.

Näiden kahden tutkimusten tulosten suora vertailu on kuitenkin hankalaa, sillä tutkimuksissa on valintaperusteet luokiteltu ja nimetty eri tavoin. Joissain kohdissa yhteys olisi suhteellisen helppo löytää ja perustella, mutta silti vertailussa olisi aivan liikaa tulkinnanvaraa. Nähdäkseni tutkimusten tulokset kuitenkin tukevat toisiaan ja antavat tämän tutkimuksen luotettavuudelle hiukan lisää pohjaa.

5.4 Tutkimuksen laajuuteen ja tavoitteisiin riittävä aineisto

Tutkimusaineisto on kerätty ohjelmistoalan ammattilaisten keskuudesta, joiden työelämää tutkittava ilmiö koskee. Tutkimukseen vastanneiden henkilöiden taustaa selvitettiin kyselytutkimuksen ensimmäisessä osassa. Tästä selvisi hyvin se, että vastaajat työskentelivät ohjelmistoalalla. Luettavuutta heikentää hieman se, että vastaajat eivät edusta kaikkia ohjelmistoalan ammattiryhmiä tasaisesti. Suurin osa eli 73 % vastaajista teki ohjelmistokehitystyötä. Ammattiryhmien jakautumista voi tarkastella kuvasta 4. Näin ollen tulokset eivät kuvaa tutkittavaa ilmiötä kaikkien ohjelmistoalan ammattilaisten näkökulmasta, vaan tulokset ovat luetettavia lähinnä kehitystyötä tekevien alan ammattilaisten kohdalla. Tutkimuksen vastaajamäärä oli vain 22 henkilöä. Muita työtehtäviä edustaa siis vain yksittäisiä vastaajia.

Tutkimuksen tuloksia on pyritty esittämään siten, että lukija pystyy päättelemään mihin tulosten analysointi on perustunut. Tämän vuoksi tuloksia esiteltäessä on käytetty suoria lainauksia ja yksityiskohtaisia tietoja vastauksista. Tuloksia on myös pyritty vertailemaan aikaisempiin tutkimustuloksiin. Näin tutkimusaineistosta on yritetty saada mahdollisimman luotettavia tuloksia. Toki luotettavuuden analysointi on viimekädessä tutkimuksen lukijoiden ja tulosten hyödyntäjien käsissä.

Kvantitatiivisen aineiston analysoinnissa tutkimustulosten luettavuuteen on syytä suhtautua varauksella. Vastaajat on rekrytoitu kyselytutkimukseen lähinnä omien verkostojen kautta. Jos sama tutkimus toistettaisiin vaikkapa satunnaisotannalla, niin ei olisi mitenkään varmaa, että samat tulokset olisivat toistettavissa. Toisaalta on kuitenkin mainittava, että kyselytutkimus oli huolellisesti suunniteltu tutkimuskirjallisuuden perusteella. Myös menetelmän valintaan oli kiinnitetty huomiota. Kyselytutkimuksen kohdalla tilastollisesti luotettavia päätelmiä ei kuitenkaan voida tehdä.

Kun vertaillaan tutkimusaineistoa ja tutkimuksen tietoperustaa, niin huomataan joidenkin asioiden olevan hieman ristiriitaisia. Vastaajilta kysyttiin, että millaisia validointiin käytettäviä ohjelmistokirjastoja he olivat työssään hyödyntäneet. Vain 43 % vastaajista oli käyttänyt JSON skeemaa hyödyntäviä ohjelmistokirjastoja. Kuitenkin Npm-sivustolla latausmäärien perusteella suosituin validointikirjasto Ajv JSON schema validator hyödyntää juuri JSON skeemaa. Tällä perusteella olisi voinut olettaa, että suurempi osa vastaajista olisi käyttänyt työssään JSON skeemoja hyödyntäviä validointikirjastoja. Ajv JSON schema validator on toki vain JavaScript ja TypeScript ohjelmointikielien ohjelmistokirjasto ja sen vuoksi tästä ei voida tehdä suoria päätelmiä tutkimuksen luotettavuudesta. Tämä kuitenkin antaa ajattelemisen aihetta.

Vaikka tutkimusaineiston laajuus on melko suppea, niin silti voidaan todeta, että kyselytutkimuksella saatu aineisto oli riittävän laaja tutkimusongelman ratkaisemiseen. Kyselytutkimus pyrittiin laatimaan niin, että kysymyksien vastauksilla voitaisiin mitata juuri tutkittavaa ilmiötä ja saataisiin valideja vastauksia tutkimuskysymyksiin. Nähdäkseni kyselytutkimuksen perusteella saatiin kvalitatiivista- ja kvantitatiivista aineistoa, joka antoi uutta tietoa tutkittavasta ilmiöstä. Tässä suhteessa tutkimuksen tavoitteet siis saavutettiin varsin hyvin.

5.5 Jatkotutkimuskohteita

Kun ajatellaan tutkittavan ilmiön laajuutta ja ohjelmistokirjastojen valintaperusteiden monimutkaisuutta, niin monia kysymyksiä jäi avoimeksi ja uusia kysymyksiä heräsi tulosten analysoinnin edetessä. Esimerkiksi kyselyyn vastasi pääasiassa ohjelmistokehityksen parissa työskenteleviä ihmisiä. Siksi kokonaiskuvaa ohjelmistoalalla työskentelevien ihmisten käsityksistä ohjelmistokirjaston ja validointikirjaston valinnassa ei syntynyt. Vastaajat olivat sen verran

suppealta osa-alueelta ohjelmistoalan ammattilaisten joukosta. Siksi asiasta voitaisiin tehdä jatkotutkimus satunnaisotannalla niin, että perusjoukkona olisi ohjelmistoalalla työskentelevät ihmiset. Perusjoukosta voitaisiin tehdä huolellinen otanta, jossa olisi edustettuna kaikkia ammattiryhmät perusjoukosta. Silloin kaikki ammattiryhmien vastaukset saataisiin edustavasti esille.

Jatkotutkimuksessa tutkittavaan ilmiöön liittyviä käsitteitä voitaisiin samalla täsmentää. Esimerkiksi asiakkaan vaatimukset ja kustannukset voidaan ymmärtää eri tavoilla riippuen vastaajan ammatillisesta taustasta riippuen. Ohjelmistoalla työskentelevä myyjä voi ymmärtää asiakkaan vaatimukset hieman eri tavalla kuin ohjelmistokehittäjä. Samoin projektijohdossa kustannukset voidaan nähdä laajemmin kuin vaikka ylläpitotyötä tekevien ihmisten parissa. Kun tarkennettaisiin kysymysten asettelua ja käsitteitä, jatkotutkimuksessa voitaisiin saada luetettavampia tuloksia.

Kyselytutkimuksen vastauksissa toistui useamman kerran ohjelmistokirjastojen käytön helppokäyttöisyys ja kehittäjämukavuus. Tästä voitaisiin johtaa useita jatkotutkimuskohteita. Ensinnäkin voitaisiin tutkia mitä asioita kehittäjämukavuuteen liitetään ohjelmistokirjastojen tai erityisesti validointiin käytettävien ohjelmistokirjastojen kohdalla. Toinen mahdollinen tutkimuskohde kehittäjämukavuudessa voisi olla se, että voidaanko helppokäyttöisellä ohjelmistokirjastolla säästää aikaa ja kustannuksia verrattuna toisiin ohjelmistokirjastoihin. Silloin helppokäyttöisyydelle voitaisiin osoittaa selkeä taloudellinen kannustin, jolloin kehittäjämukavuus olisi selkeästi perusteltavissa.

Tutkimusaineistosta tuli myös esille, että validointi mielletään kipupisteeksi erityisesti dynaamisesti tyypitetyn JavaScriptin kohdalla. Validoinnista ja dynaamisesti tyypitetyistä ohjelmointikielistä voitaisiin varmasti löytää lisää tutkittavaa. Jatkotutkimuskohteet voisivat siis liittyä vaikkapa JavaScriptin ja validoinnin haasteisiin.

5.6 Itsearviointi

Opinnäytetyön aihe on suhteellisen ajankohtainen, vaikka ohjelmistokirjastot ja validointi eivät sinällään ole mitään uutta. Ohjelmistokirjastojen valintaa ja varsinkaan validointiin käytettävien kirjastojen valintaa ei ole tutkitti kovin paljon, joten se antaa uutta näkökulmaa opinnäytetyölle. Aiheella on merkitystä oman ammatillisen kehitykseni kannalta ja yleisemminkin, sillä lähes kaikki nykyiset ohjelmistot perustuvat ohjelmistokirjastojen hyödyntämiseen. Validointi on tärkeä osa ohjelmistojen tietoturvaa ja luotettavuutta, joten siihen käytettävien ohjelmistokirjastojen valintaprosessia ei voi väheksyä.

Tutkimuksen kohde on rajattu suhteellisen hyvin ja siihen liittyviä valintoja on pyritty perustelemaan. Rajausta olisi voinut parantaa keskittymällä yhteen ohjelmointikieleen ja sen

ohjelmistokirjastoihin. Tämä olisi selkeyttänyt tutkittavaa asiaa entisestään. Esimerkiksi rajaamalla tutkimuskohde JavaScript tai TypeScript ohjelmointikieleen olisi ollut merkittävä parannus. Heikosti tyypitetyn JavaScriptin kohdalla validoinnissa on eri tavalla haasteita kuin vahvasti tyypitetyissä kielissä. JavaScript-pohjainen TypeScript on ollut viimevuosian vahvassa nousussa ja sen osuus kasvoi GitHubissa 37,8 % vuosina 2021–2021 (GitHub 2022). Tämä olisi tehnyt tutkimuksesta paremmin rajatun ja ajankohtaisemman.

Opinnäytetyön tavoite on tarjota taustatietoa validointikirjastojen valinnasta ja ohjelmistokirjastojen valinnasta ylipäätään. Se on suunnattu ammattialaan nähden oikein eli ohjelmistoalalle pyrkiville ja siellä jo työskenteleville ihmisille. Työn tavoitteet ovat perusteltuja, sillä ohjelmistokirjaston ja validointikirjaston valinta ei ole aina kovin yksiselitteistä ja helppoa. Hyvät perustiedot valintaperusteista voisivat helpottaa työtä merkittävästi. Kun pohditaan työn tarkoitusta ja sen suhdetta tietoperustaan, niin siihen olisi pitänyt saada enemmän juuri ohjelmistokirjastojen valintaa käsitteleviä lähteitä.

6 Yhteenveto

Opinnäytetyössä tutkittiin validoinnissa käytettävien ohjelmistokirjastojen valintaa ja REST-rajapintojen sekä JSON-datamuodon merkitystä siinä yhteydessä. Aihe on merkityksellinen monestakin syystä. Lähes kaikki ohjelmistot perustuvat ohjelmistokirjastoihin ja ne vaikuttavat ohjelmiston toimintavarmuuteen, tietoturvaan, suorituskykyyn ja lisensointiin. REST on taas yleinen rajapinta-arkkitehtuuri ja JSON on hyvin suosittu datamuoto. Lisäksi lähes kaikki sovellusten välillä liikkuva data tulisi validoida esimerkiksi tietoturvaongelmien välttämiseksi ja datan eheyden takaamiseksi.

Tutkimus toteutettiin kyselytutkimuksena ohjelmistoalan ammattilaisille. Kyselyyn oli mahdollista vastata noin kuukauden ajan ja vastaajia rekrytoitiin jakamalla linkkiä sosiaalisessa mediassa. Tutkimuksessa selvitettiin, että millaisia asioita huomioidaan validointikirjaston valinnassa ja eroaako se muiden ohjelmistokirjastojen valinnasta. Lisäksi tutkittiin, miten REST-rajapinta-arkkitehtuuri huomioidaan ja millainen merkitys JSON datamuodolla on validointikirjaston valinnassa.

Tutkimuksen tulosten perusteella voidaan todeta, että yleiset ohjelmistokirjastojen valintaperusteet soveltuvat yhtä hyvin validointikirjaston valintaan kuin muidenkin ohjelmistokirjaston valintaan. Erot ovat vähäisiä ja tuntuvat perustuvan ohjelmistoalan ammattilaisten mieltymyksiin painottaa eri asioita valinnassa. Validointikirjastojen valinnassa huomioidaan erityisesti yhteensopivuus, helppokäyttöisyys, skaalautuvuus ja tietoturva sekä se, että validointiin voidaan käyttää ohjelmistokehyksen ominaisuuksia.

Tulosten mukaan REST-rajapinta-arkkitehtuuri tulee ottaa huomioon validointikirjasto valittaessa. Tärkeitä näkökohtia ovat tietoturva, skaalautuvuus ja suorituskyky. Lisäksi käyttömukavuus ja validointikirjaston kyky tuottaa tietotyypit ja validointi ovat merkittäviä asioita. JSON-datamuodon merkitys validointikirjaston valintaan on hyvin pieni.

Lähteet

Akamai 2019. [state of the internet] / security Retail Attacks and API Traffic Report: Volume 5, Issue 2. Luettavissa: https://www.akamai.com/site/it/documents/state-of-the-internet/state-of-the-internet-security-retail-attacks-and-api-traffic-report-2019.pdf. Luettu 9.2.2023.

Altexsoft 2022. REST API: Key Concepts, Best Practices, and Benefits. Luettavissa: https://www.altexsoft.com/blog/rest-api-design/. Luettu. 13.2.2023

Amazon AWS. 2022. What is an API?. Luettavissa: https://aws.amazon.com/what-is/api/. Luettu: 28.11.2022.

Christensson, P. 2016. API Definition. Luettavissa: https://techterms.com/definition/api. Luettu: 17.9.2022.

Christensson, P. 2018. Schema Definition. Luettavissa: https://techterms.com/definition/schema. Luettu: 27.9.2022.

Droettboomm M. 2022. Understanding JSON Schema: What is schema? Luettavissa: https://json-schema.org/understanding-json-schema/about.html. Luettu: 1.12.2022.

DTCC. 2022. Data Validation: Business, Syntax, and Semantic Rules. Luettavissa: https://dtcclearning.com/helpfiles/data/alert/im_help/Content/Topics/set_instr_val_rules/types_of_val_rules.htm. Luettu: 28.11.2022.

Erinç, Y. 2020. The Benefits of Going RESTful – What is REST and Why You Should Learn About It. Luettavissa: https://www.freecodecamp.org/news/benefits-of-rest/. Luettu: 13.2.2023

Erlund, K. 2022. IT2022 EJT - erityisehtoja tietojärjestelmien ja asiakaskohtaisten ohjelmistojen toimituksista. Teoksessa Erlund, K., Aalto-Setälä, M., Hynönen, K., Lilja, J., Lindfors, A., Nevasalo, T., Salminen, J. & Turunen, J. (toim.) IT2022 – Käytännön käsikirja, s. 216–217. Helsingin Kamari Oy. Helsinki.

Fowler, M. 2005. InversionOfControl. Luettavissa: https://martinfowler.com/bliki/InversionOfControl.html. Luettu: 27.9.2022.

GitHub 2022. The top programming languages. Luettavissa: https://octoverse.github.com/2022/top-programming-languages. Luettu: 7.3.2023.

Honkanen, M., Moilanen, M., Niinioja, M. & Seppänen, M 2018. API-talous 101. Alma Talent Oy. Helsinki. E-kirja. Luettu: 25.9.2022.

Introducing JSON. Luettavissa: https://www.json.org/json-en.html. Luettu: 27.9.2022.

Iyer, B & Subramaniem, M. 2015. The Strategic Value of APIs. Harvard Business Review. Luettavissa: https://hbr.org/2015/01/the-strategic-value-of-apis. Luettu: 28.11.2022.

Janssen, C. & Janssen D. 2016. Software Library. Luettavissa: https://www.techopedia.com/definition/3828/software-library. Luettu: 27.9.2022.

OpenJS Foundation 2022a. JSON Schema. Luettavissa: https://json-schema.org. Luettu: 27.9.2022.

OpenJS Foundation 2022a. JSON Schema Implementations. Luettavissa: https://json-schema.org/implementations.html. Luettu: 28.9.2022.

Juviler, J. 2021. 4 Types of APIs All Marketers Should Know. Luettavissa: https://blog.hubspot.com/website/types-of-apis. Luettu: 28.11.2022.

Kulchenko, P., Snell, J. & Tidwell, D. 2001. Programming Web Services with SOAP. O'Reilly Media, Inc. Beijing, Cambridge, Farnham, Köln, Sebastopol & Tokyo. E-kirja. Luettu. 25.9.2022.

IEEE Standardi IEEE Std 1484.11.2. 2020. Standard for Learning Technology--ECMAScript Application - Programming Interface for Content to Runtime Services Communication - Redline," in IEEE Std 1484.11.2-2020 Luettavissa: https://ieeexplore.ieee.org/document/9360685. Luettu: 17.9.2022.

Marrs, T. 2017. JSON at Work. O'Reilly Media, Inc. Sebastopol. E-kirja. Luettu. 27.9.2022.

Maayan, G. 2021. What are APIs? A Complete Guide. Luettavissa: https://www.computer.org/publications/tech-news/trends/what-are-apis. Luettu: 9.2.2023

Npm, inc. 2020. Luettavissa: https://www.npmjs.com. Luettu: 29.9.2022.

NpmCompare. 2020. Luettavissa: https://npmcompare.com. Luettu: 1.12.2022.

Open Knowledge Finland ry. Open API -työryhmä 2014. Avoimen rajapinnan määritelmä. Luettavissa: http://avoinrajapinta.fi. Luettu: 23.9.2022.

OWASP Cheat Sheet Series Team. 2021. Input Validation Cheat Sheet. Luettavissa: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html. Luettu: 28.11.2022.

Phoenix, J. 2022. What is Data Validation and When Do You Do It? Luettavissa: https://understandingdata.com/what-is-data-validation-and-where-do-you-validate-data/. Luettu:27.9.2022.

Poetker, B. 2022. What Is an API? How APIs Improve Application Development. Luettavissa: https://www.g2.com/articles/what-is-an-api. Luettu: 28.11.2022.

Raj, P & Subramanian, H. 2019a. Hands-On RESTful API Design Patterns and Best Practices. Packt Publishing. Birminghamn. E-kirja. Luettu: 15.9.2022.

Sandoval, K. 2016. What Data Formats Should My API support? Luettavissa: https://nordicapis.com/what-data-formats-should-my-api-support/. Luettu: 28.11.2022.

Simpson, J. 2022. 20 Impressive API Economy Statistics. Luettavissa: https://nordicapis.com/20-impressive-api-economy-statistics/. Luettu: 29.11.2022.

The GraphQL Foundation 2022a. Introduction to GraphQL. Luettavissa: https://graphql.org/learn/. Luettu: 24.9.2022.

The GraphQL Foundation 2022b. Best Practices. Luettavissa: https://graphql.org/learn/best-practices/. Luettu: 24.9.2022.

Vargas, E., Aniche, M., Treude, C., Bruntink, M. & Gousios, G. 2020. Selecting Third-Party Libraries: The Practitioners' Perspective. Luettavissa: https://arxiv.org/pdf/2005.12574.pdf. Luettu: 1.12.2022.

Walker, A. 2022a. What is an API? Full Form, Meaning, Definition, Types & Example. Luettavissa: https://www.guru99.com/what-is-api.html. Luettu: 12.2.2022.

Walker, A. 2023b. SOAP vs REST API: Difference Between Web Services. Luettavissa: https://www.guru99.com/comparison-between-web-services.html. Luettu: 13.2.2023

Walker, A. 2022c. JSON vs XML – Difference Between Them. Luettavissa: https://www.guru99.com/json-vs-xml-difference.html. Luettu: 12.2.2022.

World Wide Web Consortium (W3C) 2015. XML essentials. Luettavissa: https://www.w3.org/standards/xml/core. Luettu. 27.9.2022.

Liitteet

Liite 1. Kyselytutkimus

Ohjelmistokirjastojen valinta – datan validointi, REST-rajapinnat ja JSON

Tämä tutkimus selvittää ohjelmistoalalla toimivien ihmisten näkemyksiä. Kysely on osa Haaga-Helian tietojenkäsittelyn tradenomikoulutuksen opinnäytetyötäni.

Kyselytutkimuksessa painotetaan validointiin käytettäviä ohjelmistokirjastoja. Niiden valinnassa kiinnitetään huomiota REST-rajapinta-arkkitehtuurin ja JSON datamuodon vaikutuksiin. Edellä mainitut asiat on valittu kyselytutkimuksen painotukseksi, koska validointia tehdään lähes jokaisessa ohjelmistossa ja REST-rajapinnat sekä JSON datamuoto ovat hyvin suosittuja nykyaikaisessa ohjelmistokehityksessä.

Kyselyyn vastaaminen vie noin 10–15 minuuttia. Kaikki vastaukset kerätään anonyymisti ja tietoja käsitellään luottamuksellisesti. Kyselytutkimukseen osallistuvia henkilöitä ei mahdollista tunnistaa tutkimuksesta.

Olen erittäin kiitollinen jokaisesta vastauksesta. Kaikki vastaukset ovat arvokkaita vaikka et olisikaan kovin perehtynyt tutkimuksessa kysyttäviin asioihin. Kyselyyn voi siis vastata hyvin matalalla kynnyksellä.

Kiitos vastauksista!

Ystävällisin terveisin,
Kimmo Lindeman, tradenomiopiskelija
kimmo.lindeman@myy.haaga-helia.fi

Seuraava

14% Valmis

1. Taustatiedot

Ensimmäisessä osiossa kysytään taustatietoja kuten työ- ja koulutustaustaa. Taustatietoja käytetään vain kyselyn vastausten analysoinnissa. Kyselyyn vastataan anonyymisti.

Ikäryhmä

- ◯ Alle 20 vuotta
- ◯ 21–30 vuotta
- ◯ 31–40 vuotta
- ◯ 41–50 vuotta
- ◯ 51–60 vuotta
- ◯ Yli 60 vuotta
- ◯ En halua vastata

Koulutusala

- ◯ Kasvatusalat
- ◯ Humanistiset ja taidealat
- ◯ Yhteiskunnalliset alat
- ◯ Kauppa, hallinto ja oikeustieteet
- ◯ Luonnontieteet
- ◯ Tietojenkäsittely ja tietoliikenne (ICT)
- ◯ Tekniikan alat (Esim. Kone- ja sähkötekniikka)
- ◯ Maa- ja metsätalousalat
- ◯ Terveys- ja hyvinvointialat
- ◯ Palvelualat
- ◯ Jokin muu, mikä? _____

Koulutusaste

- ◯ Perusaste
- ◯ Toinen aste
- ◯ Opistoaste
- ◯ Alempi korkeakouluaste ja ammatillinen korkea-aste
- ◯ Ylempi korkeakouluaste
- ◯ Tutkijakoulutusaste
- ◯ Jokin muu, mikä? _____

Työkokemus ohjelmistoalalta

- ◯ Alle 1 vuosi
- ◯ 1–2 vuotta
- ◯ 3–5 vuotta
- ◯ 6–10 vuotta
- ◯ 11–15 vuotta
- ◯ 16–20 vuotta
- ◯ Yli 20 vuotta

Ammattiasema

- ◯ Johtaja
- ◯ Toimihenkilö
- ◯ Työntekijä
- ◯ Yrittäjä
- ◯ Opiskelija
- ◯ Työtön
- ◯ Jokin muu, mikä? _____

Pääasiallinen työnkuva

- ◯ **Myyntiin tai asiakkuuksiin liittyvät tehtävät**
  Esim. myyntineuvottelija tai asiakkuuspäällikkö
- ◯ **Projekti- tai järjestelmävastuu**
  Esim. tuoteomistaja, projektipäällikkö, tai scrum-master
- ◯ **Kehitystyö**
  Esim. ohjelmistokehittäjä, ohjelmistoarkkitehti tai konsultti
- ◯ **Testaus ja laadunvarmistus**
  Esim. testaaja
- ◯ **Tukipalvelut**
  Esim. asiakastuki tai ylläpitotyöt
- ◯ **Jokin muu, mikä** _____

Edellinen Seuraava

29% Valmis

2. Ohjelmistokirjastojen valinta - Osa 1.

Tässä osiossa keskitytään ohjelmistokirjastojen valintaan ja siihen vaikuttaviin seikkoihin. Pohdi asioita omien kokemustesi valossa.

Millä perusteilla valitset ohjelmistokirjaston?

Voit valita useampia vaihtoehtoja.

- [ ] **Latausmäärä**
  Kuinka paljon ohjelmistokirjastoa on ladattu käyttämästäsi palvelusta? Esim. npm tai yarn
- [ ] **Riippuvuuksien määrä**
  Kuinka monta uutta riippuvuutta ohjelmistokirjaston mukana tulee
- [ ] **Päivitykset**
  Päivitetäänkö ohjelmakirjastoa aktiivisesti ja kuinka useita versioita siitä on julkaistu
- [ ] **Avoimet ongelmat (Bugs)**
  Kuinka paljon ohjelmistokirjastolla on avoimia ongelmia GitHubissa? Kuinka toimintavarma ja virheetön ohjelmistokirjasto on?
- [ ] **Avointen vetopyynnöt (Pull requests)**
  Kuinka paljon ohjelmistokirjastolla on avoimia vetopyyntöjä GitHubissa?
- [ ] **Tuki ja dokumentaatio**
  Onko ohjelmistokirjaston käyttöön hyvä tuki ja dokumentaatio? Löytyykö kirjastolle wikisivusto jne.
- [ ] **Kehittäjät ja yhteisö**
  Onko ohjelmistokirjaston kehittäjäyhteisö laaja ja luotettavana pidetty? Onko kirjastolla tunnettuja tukijoita?
- [ ] **Kirjaston suorituskyky**
  Kuinka nopeasti ohjelmistokirjaston suorittaa annetut tehtävät? Toiminta suuren rasituksen alla?
- [ ] **Lisensointi**
  Onko kirjasto avointa lähdekoodia? Mikä on ohjelmistokirjaston lisenssi? Esim. MIT, GNU GPL, BDS-3-Clause tai AFL v2.1.
- [ ] **Yrityksen käytännöt**
  Työnantajan toimintaohjeet ohjelmistokirjastojen käytöstä
- [ ] **Asiakkaan vaatimukset**
  Asiakasyrityksen vaatimukset ja toivomukset
- [ ] **Projektin vaatimukset**
  Projektin tekniset- tai liiketoimintavaatimukset tai muut vastaavat tekijät
- [ ] **Oma osaaminen**
  Kirjasto on minulle tuttu ja osaan hyödyntää sitä tehokkaasti. Aikaa ei kulu uuden ohjelmistokirjaston opettelemiseen.
- [ ] **Kustannukset**
  Ohjelmistokirjaston käytöstä aiheutuvat suorat ja välilliset kustannukset.
- [ ] **Jokin muu, mikä?**

Edellinen Seuraava

43% Valmis

2. Ohjelmistokirjastojen valinta - Osa 2.

Toisessa osiossa keskitytään ohjelmistokirjastojen valintaan ja siihen vaikuttaviin seikkoihin. Pohdi asioita omien kokemustesi valossa.

Arvioi eri tekijöiden merkitystä ohjelmistokirjaston valinnassa

	Merkitys				
	Erittäin tärkeä	Melko tärkeä	Ei tärkeä eikä merkityksetön	Melko merkityksetön	Täysin merkityksetön
Latausmäärä	○	○	○	○	○
Riippuvuuksien määrä	○	○	○	○	○
Päivitykset	○	○	○	○	○
Avoimet ongelmat (Bugs)	○	○	○	○	○
Avoimet vetopyynnöt (Pull requests)	○	○	○	○	○
Tuki ja dokumentaatio	○	○	○	○	○
Kehittäjät ja yhteisö	○	○	○	○	○
Kirjaston suorituskyky	○	○	○	○	○
Lisensointi	○	○	○	○	○
Yrityksen käytännöt	○	○	○	○	○
Asiakkaan vaatimukset	○	○	○	○	○
Projektin vaatimukset	○	○	○	○	○
Oma osaaminen	○	○	○	○	○
Kustannukset	○	○	○	○	○

Miten valitset ohjelmistokirjaston?

Kerro asioista, joihin kiinnität huomiota valitessasi ohjelmistokirjastoa. Kirjaa erityisesti ne asiat, joita ei ole huomioitu edellisissä kysymyksissä.

Edellinen

Seuraava

57% Valmis

3. Datan validointiin käytettävät ohjelmistokirjastot

Tämän osion aiheena on datan validointi ja siihen käytettävät ohjelmistokirjastot. Mihin asioihin kiinnität huomiota valitessasi validointiin käytettävää ohjelmistokirjastoa? Vastaa omaan kokemukseen perustuen. Jos sinulla ei ole kokemusta asiasta voit jättää vastaamatta kysymyksiin.

Millaisia validointiin käytettäviä ohjelmistokirjastoja olet käyttänyt?

Valitse kaikki vaihtoehdot, joita olet käyttänyt. Mainitse myös vaihtoehdot, joita on käytetty projekteissa, jossa olet ollut osallisena.

- ☐ JSON skeemaa hyödyntävät kirjastot
  Esim. AJV, Json-schema, jsonschema tai Schema-utils
- ☐ Objekti-skeemaa käyttävät kirjastot
  Esim. Joi, Yup, @homanwhocodes/object-schema tai Zod
- ☐ Lomakkeiden validointi
  Esim. Formik tai React-Hook-Forms
- ☐ Merkkijonon validointi ja puhdistus
  Esim. Validate tai validator.js
- ☐ Datatyypin validointi
  Esim. Type tai typechecker
- ☐ Yksittäisen tiedon validointi
  Esim. Email-validator, Isemail, tai Finnish-ssn
- ☐ En ole käyttänyt
- ☐ Jokin muu, mikä? [ ]

Mihin asioihin kiinnität huomiota validointiin käytettävän ohjelmistokirjaston valinnassa?

Kerro validointikirjaston valinnasta vapaamuotoisesti. Mainitse erityisesti ne asiat, joihin kiinnität huomiota verrattuna jonkin muun ohjelmistokirjaston valintaan.

[ ]

Kuinka paljon mielestäsi validointiin käytettävän kirjaston valinta eroaa muiden ohjelmistokirjastojen valinnasta?

	Eroaa täysin	Paljon	Jonkin verran	En osaa sanoa	Vähän	Ei lainkaan
Kuinka paljon eroaa?	○	○	○	○	○	○

Kerro eroavaisuuksista

Edellisen kysymyksen vastausten perusteella kerro validointiin käytettävän kirjaston ja muiden muiden ohjelmistokirjastojen valinnan eroista.

[ ]

Edellinen Seuraava

71% Valmis

4. REST-rajapinnat ja validointikirjastot

Neljännessä osioissa kysytään REST-rajapinta-arkkitehtuurin merkitystä validointikirjaston valinnassa. Vastaa omaan kokemukseen perustuen. Jos sinulla ei ole kokemusta asiasta voit jättää vastaamatta kysymyksiin.

Miten huomioit REST-rajapinta-arkkitehtuurin validointikirjaston valinnassa?

Merkityksellisiä asioita ovat esimerkiksi suorituskyky, virheettömyys, skaalautuvuus, tietoturva tai mitä tahansa asia, jonka näet olevan tärkeä. Kerro asiasta omiin kokemuksiisi nojaten.

Arvioi REST-rajapinta-arkkitehtuurin vaikutusta validointikirjaston valintaan

	Erittäin tärkeä	Melko tärkeä	Ei tärkeä eikä merkityksetön	Melko merkityksetön	Täysin merkityksetön
REST-rajapinta-arkkitehtuuri	○	○	○	○	○

Edellinen Seuraava

86% Valmis

5. JSON-datamuoto ja validointikirjastot

Viimeisessä osiossa selvitetään JSON-datamuodon merkitystä validointikirjastojen valintaan. Vastaa omaan kokemukseen perustuen. Jos sinulla ei ole kokemusta asiasta voit jättää vastaamatta kysymyksiin.

Miten huomioit JSON-datamuodon validointikirjaston valinnassa?

Asettaako JSON-datamuoto erityisvaatimuksia validointikirjastolle? Kerro asiasta omiin kokemuksiin nojaten.

Arvioi JSON datamuodon merkitystä validointikirjaston valintaan

	Erittäin tärkeä	Melko tärkeä	Ei tärkeä eikä merkityksetön	Melko merkityksetön	Täysin merkityksetön
JSON	○	○	○	○	○

Edellinen Lähetä

100% Valmis