Tampereen yliopisto

Mikko Tuominen

REACT.JS JA VUE.JS YKSISIVUISTEN VERKKOSOVELLUSTEN KEHITYKSESSÄ

Käyttöliittymäkehykset vertailussa

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Kesäkuu 2023

TIIVISTELMÄ

Mikko Tuominen: React.js & Vue.js yksisivuisten verkkosovellusten kehityksessä –
Käyttöliittymäkehykset vertailussa

Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Kesäkuussa 2023

Verkkosovellusten yleistyessä yksisivuiset verkkosovellukset (eng. Single-page Application) ovat kasvattaneet suosiotaan niiden tarjoaman nopeuden ja natiivisovelluksen kaltaisen käyttäjäkokemuksen ansiosta. Näiden sovellusten frontendin kehittämisessä käytetään yleisesti JavaScript-pohjaisia avoimen lähdekoodin sovelluskehyksiä, jotka tarjoavat kehittäjälle valmista ohjelmakoodia ja rakenteita, joita käyttämällä kehitys- ja ylläpitoprosessia voidaan nopeuttaa ja helpottaa. Viimeisen 10 vuoden aikana tällaisia sovelluskehyksiä on ilmaantunut monia, ja sopivan kehyksen valinta saattaa osoittautua haasteelliseksi.

Tässä tutkielmassa tarkastellaan kirjallisuuskatsauksen kautta yksisivuisia verkkosovelluksia ja esitellään kaksi niiden kehittämisessä suosittua sovelluskehystä, Facebookin kehittäneen Metan luoma React.js ja Googlelle työskennelleen Evan Youn kehittämä Vue.js. Tutkielma kuvailee yksisivuisia verkkosovelluksia, Reactin ja Vuen taustoja ja perusominaisuuksia ja lopuksi tarkastelee niillä toteutettujen testisovelluksien teknistä suorituskykyä vertailleita tutkimuksia. Tutkielman tavoitteena on antaa verkkosovelluskehittäjälle näkökulmia auttamaan sovelluskehyksen valinnassa.

Tutkielmassa havaitaan, että kehykset ovat rakenteeltaan ja ohjelmointityyliltään hyvin samankaltaisia, Vuen tarjotessa laajemman kokonaisuuden ominaisuuksia verrattuna lisäkirjastoja tarvitsevaan Reactiin. Reactilla ja Vuella toteutettujen verkkosovellusten suorituskykyä vertailleissa tutkimuksissa havaittiin pieniä eroja eri mittareilla molempien hyväksi, mutta kokonaisuutena erot eivät olleet huomattavia. Havaittuja eroja voidaan pitää merkittävinä vain, jos kehitetyn sovelluksen kannalta jonkin tietyn ominaisuuden olisi erityisen tärkeää olla mahdollisimman tehokas. Tämän takia tutkielma kehottaa sovelluskehyksen valinnassa ottamaan huomioon myös projektiin osallistuvien kehittäjien mieltymykset ja ohjelmointitaustan.

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla

Avainsanat: Yksisivuinen verkkosovellus, sovelluskehys, React.js, Vue.js

SISÄLLYSLUETTELO

Lyhenteet ja merkinnät

AJAX	Asynchronous JavaScript and XML
Aika interaktiivisuuteen	Time to Interactive
API	Application Programming Interface
DOM	Dokumenttiobjektimalli
Eriyttämisen periaate	Separation of Concerns
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
JSX	JavaScript XML
Nostaminen	Hoisting
Sulautettu	Embedded
Templaatti	Template
Viiksisyntaksi	Moustache Syntax

1 Johdanto

Verkkosivuista on tullut oletusarvoisen tärkeä toiminta-alusta lähes kaikille organisaatioille. Samalla käyttäjien odotukset verkkosivujen käytettävyydestä ja responsiivisuudesta ovat lisääntyneet. Yksisivuiset verkkosovellukset ovat kasvattaneet suosiotaan näihin vaatimuksiin hyvin sopivien ominaisuuksiensa takia. Verrattuna perinteisempiin monisivuisiin sovelluksiin, ne ovat suorituskykyisempiä ja mahdollistavat nopeutensa ja reaktiivisuutensa ansiosta natiivisovelluksen kaltaisen käyttäjäkokemuksen luomisen. Tällaisten sovellusten käyttöliittymien kehittämisessä on yleistä käyttää erilaisia JavaScript -sovelluskehyksiä, jotka sisältävät valmiiksi koodattuja ohjelman osia ja työkaluja sovelluksen kehitys- ja ylläpitoprosessin tehostamiseksi (Edwin, 2014).

Verkkosovellusten käyttöliittymien kehittämiseen käytetyistä sovelluskehyksistä suosituin on Facebookin kehittämisestä tunnetun Metan luoma avoimen lähdekoodin käyttöliittymäkirjasto React.js (Greif & Burel, 2022; Stack Overflow, 2022). Toiseksi käytetyin, mutta toisaalta kehittäjien keskuudessa vähemmän pidetty kehys on Googlen kehittämä Angular (Greif & Burel, 2022). Sen haastajaksi on hiljattain noussut suosiossa tasaisesti kasvanut Vue.js, joka taas on kehittäjien keskuudessa lähes yhtä pidetty kuin React (Stack Overflow, 2022). Vuen suosio kasvu ja hyvä maine tekevät siitä mielenkiintoisen vertailukohteen markkinaa selvästi johtavalle Reactille. Molemmat sovelluskehykset myös käyttävät virtuaalista dokumenttiobjektimallia, mikä parantaa niiden teknisen tehokkuuden vertailukelpoisuutta (Novac et al., 2021).

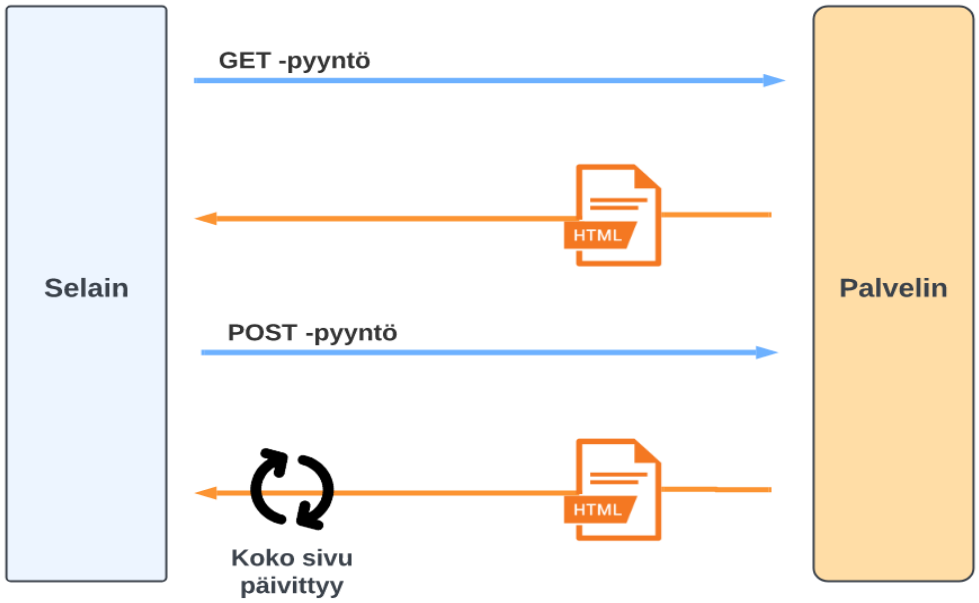
Valinta useiden ominaisuuksiltaan hyvin samankaltaisilta vaikuttavien sovelluskehysten välillä voi olla haastavaa, etenkin jos kehykset eivät ole entuudestaan kehittäjälle tuttuja. Tämän tutkielman tavoite on avata lukijalle mitä sovelluskehykset tarjoavat ja minkälaisiin käyttötapauksiin React ja Vue voisivat sopia. Tutkielman taustoitus alkaa yksisivuisten web-sovellusten rakenteen ja kyvykkyyksien tarkastelulla luvussa kaksi. Tämän jälkeen tarkastellaan, minkälaisia ydinominaisuuksia kahdella sovelluskehyksellä on. Lopuksi tarkastellaan, miten niillä toteutetut sovellukset vertautuvat toisiinsa.

Tiedonhaussa lähteitä rajaavaksi vuodeksi valittiin 2013, jolloin React julkaistiin.

2 Modernit verkkosovellukset

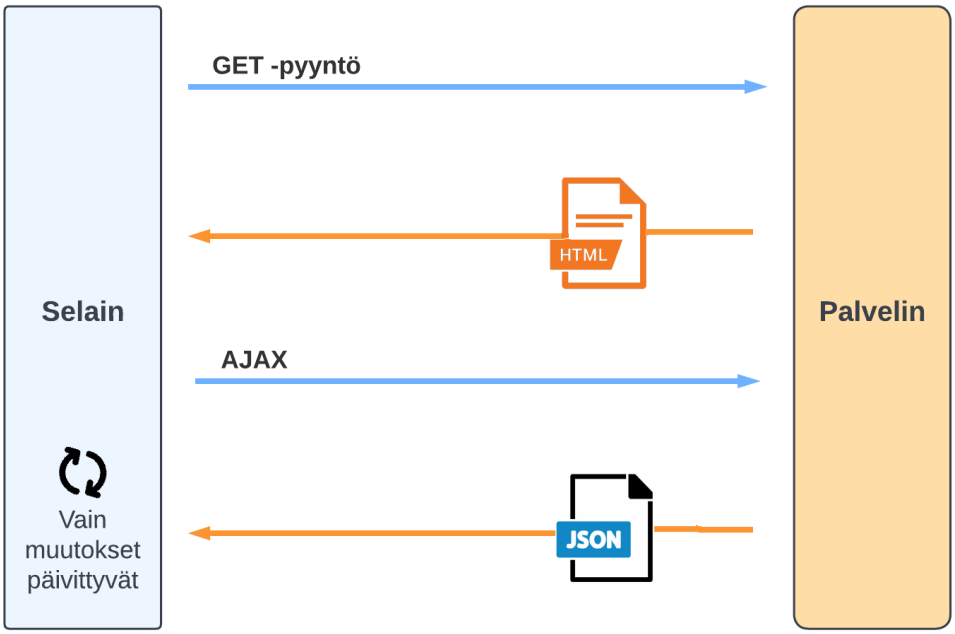
Modernien verkkosovellusten kehittämisessä käytetään yleisesti kahta lähestymistapaa. Monisivuisten verkkosovellusten mallissa suurin osa sovelluslogiikasta tapahtuu palvelimella. Uudemmassa yksisivuisten sovellusten mallissa käyttöliittymälogiikka tapahtuu selaimessa, palvelimen kanssa erilaisten ohjelmointirajapintojen eli API:en (engl. Application Programming Interface) avulla kommunikoiden (Smith, 2022). Myös molempia lähestymistapoja yhdistelevän hybridimallin käyttäminen on mahdollista. Esimerkiksi monisivuisen verkkosovelluksen sisälle voidaan näin rakentaa yksisivuisen sovelluksen kaltaisia pieniä alisovelluksia (Smith, 2022).

Monisivuisissa verkkosovelluksissa jokainen pyyntö uudesta näkymästä tai käyttäjän niihin tekemistä muutoksista muodostetaan HTML-sivuna palvelimella ja vaatii koko sivun uudelleenpäivityksen (Scott, 2016). Kuva 1 havainnollistaa monisivuisen verkkosovelluksen toimintaa. Käyttäjän alkuperäiseltä sivulta tekemän POST-pyynnön jälkeen palvelin lähettää selaimeen uuden HTML-sivun. Koko sivu on päivitettävä uudelleen, jotta uusi sivu näkyy käyttäjälle selaimessa.



Kuva 1 Monisivuisen verkkosovelluksen toiminta

Yksisivuisessa verkkosovelluksessa koko sovellus koostuu yhdestä HTLM-sivusta, joka ladataan vain kerran. Tämän sivun näkymiä tai niiden osia selain manipuloi ohjelmakoodin perusteella, (Mozilla, 2023a). Sovellus hakee käyttämänsä datan tarpeen vaatiessa palvelimelta AJAX-tekniikkaa hyödyntäen (Asynchronous JavaScript and XML). Data yhdistetään HTML:ään dynaamisesti selaimessa (Scott, 2016). Kuva 2 havainnollistaa yksisivuisen verkkosovelluksen toimintaa, jossa ensimmäisen latauksen jälkeen palvelimelta haetaan vain JSON-muotoista (JavaScript Object Notation) dataa, jota selain voi lisätä näkymiin päivittämättä koko sivua.



Kuva 2 Yksisivuisen verkkosovelluksen yksinkertaistettu toimintaperiaate

Yksisivuiset verkkosovellukset pystyvät dynaamisesti päivittämään osia sovelluksen näkymistä, ja muutokset näkyvät käyttäjälle välittömästi. Ensimmäisen sivulatauksen jälkeen serverin ja selaimen välinen kommunikaatio on tahdistamatonta ja vain kevyestä datasta koostuvaa, mikä tekee yksisivuisten verkkosovellusten tiedonsiirrosta erittäin sujuvaa. Tämä yhdistettynä kykyyn renderöidä muutoksia suoraan näkymiin ilman uusia sivulatauksia luo perinteisiin verkkosovelluksiin verrattuna sulavan ja häiriövapaan käyttäjäkokemuksen. (Scott, 2016)

3 Sovelluskehykset

Sovelluskehys on integroitu kokoelma keskenään työskenteleviä valmiita ohjelmakomponentteja, joita käytetään valmiin ohjelmistoarkkitehtuurin tuottamiseen samantyyppisissä sovelluksissa, kuten tämän tutkielman tapauksissa verkkosovelluksissa (Edwin, 2014). Kehittäjä voi tarpeen vaatiessa muokata kehysten tarjoamia ratkaisuja paremmin senhetkistä käyttötapausta palveleviksi. Sovelluskehysten tavoitteena on yksinkertaistaa kehitysympäristöä ja antaa kehittäjien keskittyä projektille asetettujen päätavoitteiden saavuttamiseen geneerisen ohjelmakoodin kirjoittamisen sijaan (Janssen, n.d.).

Tässä tutkielmassa vertailluista sovelluskehyksistä Vue on selkeästi määritelty sovelluskehykseksi, mutta React on kehittäjiensä mukaan ohjelmointikirjasto. Ohjelmointikirjastot ovat kehyksiin verrattuna kevyempi kokoelma jotain tiettyä tehtävää varten kirjoitettua valmista ohjelmakoodia, joka ei kuitenkaan määritä koko ohjelman rakennetta. Kirjallisuudessa ja alan keskusteluissa Reactiin kuitenkin usein viitataan sovelluskehyksenä, ja sitä käsitellään rinnakkain sovelluskehyksien kuten Vuen ja Angularin kanssa. Reactia myös voidaan käyttää sovelluskehyksenomaisesti, koko ohjelman rakennetta alusta lähtien määrittävänä tekijänä. Näiden syiden takia tässä tutkielmassa Reactia tarkastellaan sovelluskehyksenä.

3.1 React.js

React.js tai React on yhdysvaltalaiselle Metalle työskennelleen Jordan Walken kehittämä, avoimen lähdekoodin JavaScript-kirjasto verkkosovellusten käyttöliittymien rakentamiseen. React kehitettiin alkujaan Facebookin käyttöliittymän kehitysprosessin yhtenäistämiseksi ja tehostamiseksi, mutta se päätettiin vapauttaa avoimeen käyttöön toukokuussa 2013 Metan ostaman Instagramin kehittäjien kiinnostuttua uudesta teknologiasta (Hámori, 2022). Meta kuvailee Reactia deklaratiiviseksi, tehokkaaksi ja helpoksi oppia, ja tarjoaa sivuillaan resursseja sovelluskehykseen tutustumiseen (React.js, 2023).

React on suosituin frontend-kehitykseen käytetyistä kirjastoista ja ohjelmointikehyksistä. Ohjelmointikeskustelufoorumi Stack Overflown kehittäjäkyselyssä 46 % vastaajista kertoi käyttäneensä sitä työssään kuluneen vuoden aikana (Stack Overflow, 2022), ja State of JS 2022 kyselyssä 81.8 % kehittäjistä kertoi joskus käyttäneensä sitä (Greif & Burel,

2022). Reactin kehittäjäyhteisö on hyvin aktiivinen, esimerkiksi tutkielman kirjoitushetkellä 28.5.2023 Stack Overflowissa Reactia koskevia keskusteluja oli 446 337, ja Google löytää React-ohjelmointia käsitteleviä tutoriaalivideoita yli 10 miljoonaa kappaletta.

Versionhallintasivusto GitHubissa Reactille on annettu 200 581 tähteä. Ohjelmistokehittäjät antavat GitHubissa tähtiä käyttämilleen projekteille, joista pitävät, ovat kiinnostuneita tai joiden muutoksia haluavat pitää silmällä. Tähdet ovat hyvä mittari, kun arvioidaan eri projektien kiinnostavuutta kehittäjäyhteisössä. Monet kehittäjät myös käyttävät projektin saamien tähtien määrää arvioidessaan, haluavatko hyödyntää projektia omassa työssään. (Borges & Tulio Valente, 2018)

3.1.1 Reactin virtuaalinen dokumenttiobjektimalli

Verkkosovelluksen tilan muuttuessa sen käyttöliittymän rakennetta kuvaavaa dokumenttiobjektimallia on manipuloitava vastaamaan tapahtunutta muutosta. Dokumenttiobjektimallin suora manipuloiminen voi olla tehotonta ja johtaa suorituskykyongelmiin etenkin monimutkaisten ja dynaamisten verkkosivujen tapauksessa. Virtuaalisen dokumenttiobjektimallin konsepti vastaa tähän haasteeseen. Luomalla muistiin oikean DOM:in virtuaalisen vastineen, v-solmuista koostuvan virtuaalisen DOM-puun, tilassa tapahtuneet muutokset voidaan ensin tehdä virtuaaliseen DOM:iin ilman oikean DOM:in suoraa manipuloimista. (Kumar, 2024)

React hyödyntää tätä konseptia, jonka avulla sovellus seuraa, pitääkö jokin käyttöliittymän komponentti renderöidä uudelleen (React.js, 2023). Havaitessaan sovelluksen tilassa tapahtuneen muutoksen, React luo uutta tilaa vastaavaan virtuaalisen DOM-puun, jota verrataan vanhaa tilaa vastaavaan puuhun tilojen välisten erojen havaitsemiseksi (React.js, 2023). Tämän jälkeen React päättelee pienimmät mahdolliset muutokset, joiden avulla oikea DOM saadaan vastaamaan uutta tilaa ja lopulta toteuttaa muutokset (React.js, 2023). Virtuaalinen dokumenttiobjektimalli tekee React-sovelluksista nopeita ja vähemmän laskentatehoa käyttäviä. Toisaalta ne kuluttavat enemmän muistia virtuaalisen mallin säilömiseen käytetyn muistin takia (Kumar, 2024).

3.1.2 Syntaksi ja komponentit

React-ohjelmoinnissa käytetään Metan JavaScriptiin kehittämää deklaratiivista JavaScript XML eli JSX-syntaksilaajennusta, joka on yhdistelmä JavaScriptiä ja HTML-

tyylistä syntaksia. JSX:n tavoitteena on tuoda yhteen sovelluksen renderöinti- ja UI-logiikka, suoraviivaistaen HTML-elementtien käsittelyä ja tehden ohjelmakoodista helpompilukuisempaa verrattuna puhtaaseen JavaScriptiin. Kuvassa 3 on Reactin oma esimerkki JSX-elementistä, jossa käytetään HTML-tyylistä `<h1>`-tagia ja aaltosulkeiden avulla sulautettua (engl. embedded) JavaScriptiä. (React.js, 2023)

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;
```

Kuva 3 JSX-elementti (React.js, 2023)

React-sovellukset koostuvat itsenäisistä komponenteista, joita on helppo käyttää uudelleen ja käyttää sisäkkäin. Modulaarisuus selkeyttää sovelluksen rakennetta ja helpottaa ylläpidettävyyttä. Kuvassa 4 esitellään funktionaalinen `Welcome`-komponentti.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));
const element = <Welcome name="Sara" />;
root.render(element);
```

Kuva 4 Yksinkertainen react-komponentti (React.js, 2023)

Komponentti saa attribuuttina `props`-objektin, joka sisältää avain-arvo -parin `name` ja lopulta palauttaa JSX-lauseen jonka sisältö muuttuu funktion parametrien mukaan. `root.render()`-metodia kutsuttaessa komponentti `<Welcome name="Sara" />` renderöityy HTML muodossa `<h1>Hello, Sara</h1>`.

3.1.3 Haasteet

Koska React on pääasiassa Model-View-Controller-mallia toteuttavien käyttöliittymien sovelluksen View-tason kehittämiseen tarkoitettu kirjasto, se ei sisällä työkaluja mallin muiden osien kehittämiseen. Käyttöliittymän muiden ominaisuuksien kehittämiseen on osattava valita ja käyttää muita kirjastoja, kuten esimerkiksi Redux-kirjastoa ohjelman tilanhallinnassa. Useiden kirjastojen käyttäminen myös lisää sovelluksen riippuvuuksia,

mikä saattaa aiheuttaa ongelmia sovelluksen pitkän aikavälin ylläpidossa kirjastojen välisten yhteensopivuuksien mahdollisesti muuttuessa ajan kuluessa.

3.2 Vue.js

Vue.js on käyttöliittymäohjelmointia varten kehitetty avoimen lähdekoodin progressiivinen sovelluskehys. Vuen ensimmäisen version kehitti Evan You Googlella AngularJS:n parissa työskennellessään. Hän halusi kehittää Angularia paremmin prototyypittämiseen soveltuvan, nopean ja kevyen sovelluskehyksen. Kehyksen ensimmäinen versio julkaistiin avoimeen käyttöön vuonna 2014. Kehitystyötä jatkaa ryhmä täysipäiväisiä ja vapaaehtoisia ohjelmistokehittäjiä, yksityisten tahojen sponsoroimana. (Vuejs.org, 2023)

Vuen kehittäjien tavoitteena on luoda verkkokehittäjien vaihteleviin tarpeisiin soveltuva joustava ja asteittain adoptoitavissa oleva progressiivinen sovelluskehys, joka on helppo integroida myös olemassa oleviin projekteihin. Vaikka käyttöliittymän View-tason kehitys onkin Vuen ytimessä, se pyrkii myös tarjoamaan kattavan työkalupakin käyttöliittymän kaikkien osien kehitystyöhön. Vue pyrkii olemaan hyödyllinen ja helposti lähestyttävä niin aloitteleville ohjelmoijille kuin kokeneemmille ammattilaisille. (Vuejs.org, 2023)

Vuen kehittäjäyhteisö on aktiivinen, vaikka keskustelu ei määrällisesti nouse aivan Reactin tasolle. Tämän työn kirjoitushetkellä 28.5.2023 Vue-ohjelmointiin liittyviä kysymyksiä oli StackOverflowissa 102 144 kappaletta, ja Google löytää 2,1 miljoonaa Vuen käyttöön liittyvää tutoriaalivideota. GitHub-tähtiä Vue:lla on 203 165, ja tähtiä seuraavan Star History-palvelun mukaan vuosina 2018–2022 tähtiä oli enemmän kuin Reactilla. Myös Vue tarjoaa sivuillaan kattavasti resursseja sovelluskehykseen tutustumiseen.

3.2.1 Kääntäjän informoima virtuaalinen dokumenttiobjektimalli

Vue käyttää DOM:in hallintaan kääntäjän informoimaa virtuaalista DOM:ia (Compiler-Informed Virtual DOM). Malli on perusperiaatteeltaan hyvin samankaltainen kuin muidenkin virtuaalista DOM:ia käyttävien sovelluskehysten. Sen toimintaa on kuitenkin pyritty tehostamaan vähentämällä sovelluksen suorituksen aikana tehtyjä vertailuja virtuaa-

listen mallien välillä. Vuen kääntäjä tulkitsee templaattia käännösvaiheessa ja jättää suoritettavaan ohjelmakoodiin ”vihjeitä”, joiden avulla vertailun kannalta epäoleelliset, esimerkiksi staattiset osat voidaan jättää vertailun ulkopuolelle. (Vuejs.org, 2023)

Kuvassa 5 on esimerkki tilanteesta, jossa kääntäjä havaitsee templaatin kahden ensimmäisen elementin sisältävän ainoastaan staattista sisältöä ja nostaa (engl. hoist) niiden v-solmujen luontikutsut pois renderöintifunktiosta. Jatkossa seuraavilla renderöintikerralla käytettäisiin aikaisemmin luotua v-solmua uusien luomisen sijaan. (Vuejs.org, 2023)

```template
<div>
  <div>foo</div> <!-- hoisted -->
  <div>bar</div> <!-- hoisted -->
  <div>{{ dynamic }}</div>
</div>
```

Kuva 5 Vue-templaatti, josta kääntäjä havaitsee staattisen ja dynaamisen sisällön (Vuejs.org, 2023)

Renderöijän havaitessa v-solmun olevan sama kuin ensimmäisessä virtuaalipuussa, se tehostaa prosessia lopettamalla niiden vertailemisen kokonaan. Jos tällaisia solmuja on ketjussa useita, renderöijä voi edelleen tehostaa prosessia yhdistämällä ne yhdeksi suureksi, staattiseksi v-solmuksi. (Vuejs.org, 2023)

3.2.2 Syntaksi ja komponentit

Vue käyttää omaa laajennettua HTML-pohjaista syntaksia, mikä mahdollistaa komponenttien rakenteen ja toiminnallisuuden kuvailun deklaratiivisella tavalla (Vuejs.org, 2023). Myös Reactin JSX:n käyttäminen on mahdollista Vuen omalla babel-pluginilla (Vuejs.org, 2019). Tukemalla myös JSX:ää Vue tarjoaa kehittäjälle mahdollisuuden käyttää sitä lähestymistapaa, josta eniten pitää, tai jonka kokee parhaiten sopivan projektin vaatimuksiin. Vuen syntaksia voisi kuvailla intuitiiviseksi ja helposti lähestyttäväksi kehittäjälle, jolle HTML:n käyttö on tuttua.

Vuessa JavaScriptiä voidaan lisätä HTML-templaattiin käyttämällä interpoloitua tekstiä, joka merkitään kaksinkertaisilla aaltosulkeilla {{ … }} eli ”viiksisyntaksilla” (engl. ”mustache syntax”). Sulkeiden sisältö tulkitaan ja suoritetaan JavaScriptina. Elementteihin voidaan myös lisätä toiminnallisuutta käyttämällä Vuelle ominaisia v-etuliitteellä merkittyjä attribuutteja eli v-direktiivejä. Esimerkkejä tällaisista direktiiveistä ovat kon-

ditionaaliseen renderöintiin käytettävä `v-if`, listan elementtien pohjalta useiden elementtien renderöimiseen käytettävä `v-for` ja tapahtumakuuntelijan lisäävä `v-on`. Ohjelmoija voi myös halutessaan luoda omia direktiivejään. (Vuejs.org, 2023)

Kuvassa 6 on esimerkki tavasta miten `v-on:click` ja `v-if="visible"` -direktiivejä ja viiksisyntaksia voidaan käyttää komponentin sisällön muuttamiseen. <p>-elementissä olevan `v-if="visible"` direktiivin takia elementti renderöidään vain, jos `visible`-arvo on tosi. Button-elementtiin sidotaan klikkaukseen reagoiva tapahtumakuuntelija `v-on:click` direktiivillä, joka klikattaessa kutsuu `toggleVisibility` -metodia. Metodi muuttaa `visible`-muuttujan arvon käänteiseksi. Havaitessaan muutoksen komponentin tilassa, Vue uudelleenrenderöi komponentin vastaamaan uutta tilaa.

```
<template>
  <div>
    <button v-on:click="toggleVisibility">Toggle</button>
    <p v-if="visible">{{ message }}</p>
  </div>
</template>

<script>
export default {
  data() {
    return {
      visible: false,
      message: 'Hello there!',
    };
  },
  methods: {
    toggleVisibility() {
      this.visible = !this.visible;
    },
  },
};
</script>
```

Kuva 6 Vue-komponentti jossa käytetään v-direktiivejä ja dynaamista sisältöä

Vuella toteutetut käyttöliittymät koostuva itsenäisistä yhden tiedoston (Single-File) komponenteista, joihin sisältyy rakennetta määrittelevä HTML-tyylinen templaatti `<template>`, JavaScriptiä sisältävä komponenttilogiikan määrittelevä `<script>` ja komponentin tyyliä määrittelevä `<style>`. Komponentit on suunniteltu olemaan modulaarisia ja uudelleenkäytettäviä, mahdollistaen käyttöliittymän osien jakamisen pienempiin kokonaisuuksiin. Tämä jaottelu noudattaa ongelmien eriyttämisen periaatetta

(Separation of Concerns), mikä parantaa komponenttien uudelleenkäytettävyyttä, luettavuutta ja ylläpidettävyyttä. (Vue.org, 2023)

3.2.3 Haasteet

Usein samanlaisten ongelmien parissa työskenteleviltä kehittäjiltä saatu tuki on arvokasta ongelmatilanteissa. Kiinalaiset suuryritykset kuten Alibaba, Tencent, Xiaomi ja Baidu käyttävät Vuea, ja Vuen kiinalainen kehittäjäyhteistö on hyvin aktiivinen (You, 2016). Paikallisen suosion seurauksena suurin osa tämän kehittäjäyhteisön tuottamasta Vueen liittyvästä sisällöstä, kuten yhteisökeskusteluista, ongelmien ratkaisuista ja tutoriaaleista on saatavilla ainoastaan kiinan kielellä. Englanninkielinen keskustelu on lisääntynyt Vuen kasvattaessa suosiotaan, mutta orgaanisen yhteisösisällön jääminen kielimuurin taakse on silti huomionarvoinen haaste.

4 Sovelluskehyksien vertailu

Vertailemalla Reactilla ja Vuella toteutettujen, samankaltaista toiminnallisuutta toteuttavien sovelluksien suorituskykyä voidaan havainnoida niiden eroja ja mahdollista sopivuutta erilaisiin käyttötapauksiin. Tähän tutkielmaan on valikoitu kaksi tällaista vertailukoetta toteuttanutta tutkimusta.

Figueiredoy et al. 2022 tutkimuksessa vertailtiin kolmella suosituimmalla ohjelmointikehyksellä ja puhtaalla JavaScriptillä toteutetun ohjelman DOM manipulaatioaikaa, kimppukokoa ja aikaa interaktiivisuuteen. Vaikka tämä tutkielma keskittyy sovelluskehysten vertailuun, vertailumielessä on myös hyvä tarkastella pelkällä JavaScriptillä toteutetun sovelluksen teknistä suoriutumista. Figueiredoy tutkimuksessa on huomattava, että testissä oli myös mukana inkrementaalista dokumenttiobjektimallia käyttävä Angular.js, joten testit eivät välttämättä keskity testaamaan Reactin ja Vuen virtuaalisen dokumenttiobjektimallin ominaisuuksia parhaalla mahdollisella tavalla.

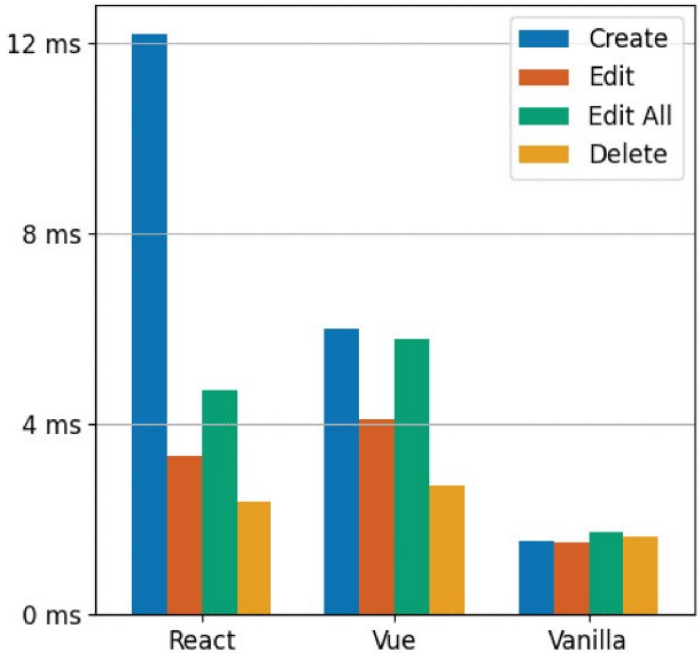
Novac et al. 2021 tutkimuksessa vertailtiin kehyksillä toteutettujen sovellusten renderöinti- ja uudelleenrenderöintiaikaa API-kutsua tehtäessä, sekä pelkän API-kutsun onnistuneeseen toteuttamiseen kulunutta aikaa. Tutkimuksen tekijät huomioivat, että kokeet suoritettiin pienillä prototyyppisovelluksilla lokaalisti, eivätkä tulokset välttämättä ole verrannollisia suurempien sovelluksien suorituskykyyn.

4.1 DOM-manipulaatioaika

Dokumenttiobjektimallin manipuloiminen on keskeisimpiä dynaamisen yksisivuisen verkkosovelluksen tehtäviä. Sovelluksen tilan muutoksien mukaan komponentteja voidaan lisätä, päivittää tai poistaa kokonaan. Onkin mielekästä vertailla, miten nopeasti eri kehyksiä käyttävät sovellukset suoriutuvat näistä tehtävistä. Sovelluskehykset tarjoavat komponenteille paikallisen elinkaarihookin (native lifecycle hook), joka erottaa komponenttien luomisen, päivittämisen ja poistamisen omiksi metodeikseen. Nopeampi elinkaari tarkoittaa nopeampaa renderöitymistä selaimessa (Figueiredoy et al., 2022).

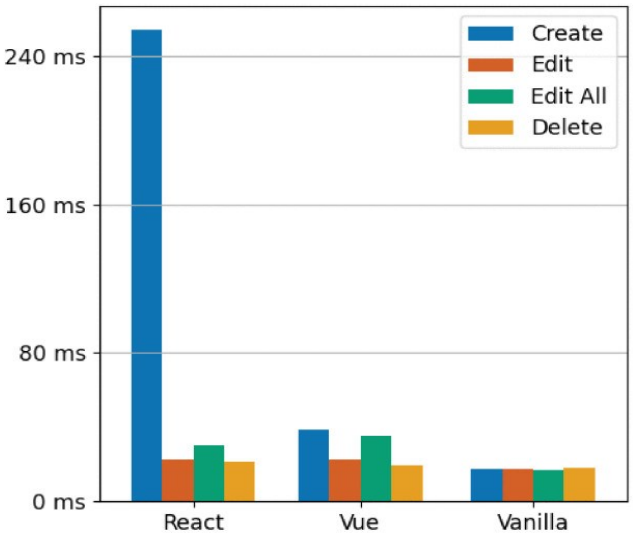
Taulukko 1 esittää 1000 HTML-elementin samanaikaiseen eri tavoilla manipuloimiseen kulunutta aikaa. Tässä skenaariossa Vue menestyy keskimäärin paremmin Reactiin verrattuna, ollen silti selvästi hitaampi kuin puhdas JavaScript-sovellus.

Taulukko 1 1000 elementin samanaikainen manipulointi (Figueiredoy et al., 2022)



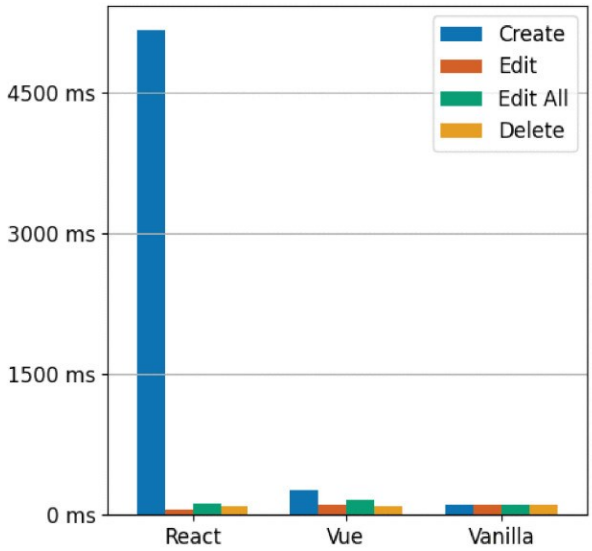
Taulukossa 2 manipuloitavien elementtien määrä on kasvatettu kymmeneen tuhanteen. Elementtien määrän kasvaessa Reactin Create-fuktion hitaus korostuu entisestään, vieden keskimäärin 254,10 ms. Vuen funktion nopeus taas pysyy suunnilleen yhtä tehokkaana kuin aikaisemmilla elementtimäärillä, vieden keskimäärin 36,5 ms. (Figueiredoy et al., 2022)

Taulukko 2 10 000 elementin samanaikainen manipulointi (Figueiredoy et al., 2022)



Lopulta kaikkein raskaimmassa skenaariossa, elementtien määrä nousee 50 000 elementtiin, jolloin elementtien erittäin suuri määrä vaikuttaa ohjelman suorituskykyyn jo yleisellä tasolla. Reactin Create-funktio vie jo 5165 ms kaikkien elementtien renderöimiseen, Vuen edelleen säilyttäessä samankaltaisen tehokkuuden kuin aikaisemmissa skenaarioissa. (Figueiredoy et al., 2022)

Taulukko 3 50 000 elementin samanaikainen manipulointi (Figueiredoy et al., 2022)

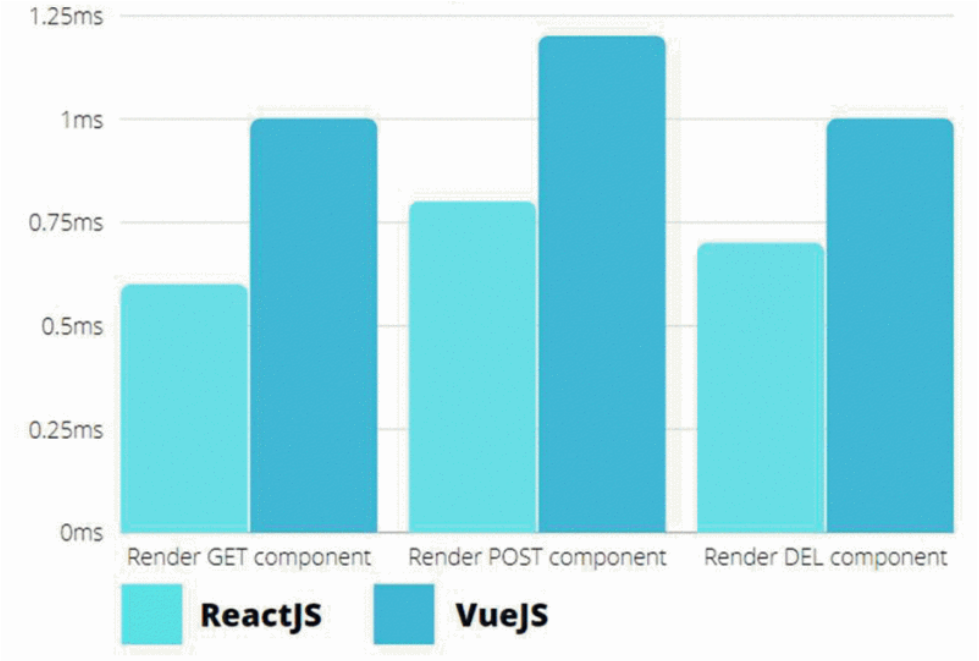


Kokeiden perusteella Vue vaikuttaa suoriutuvan tasaisesti kaikilla elementtimäärillä, kun taas Reactin Create-funktio hidastuu eksponentiaalisesti elementtien määrän kasvaessa hyvin suureksi. Käyttötapauksissa, joissa elementtejä on manipuloitava paljon, Create-funktion hidastuminen olisi syytä ottaa huomioon. (Figueiredoy et al., 2022).

4.2 Renderöinti ja API-kutsut

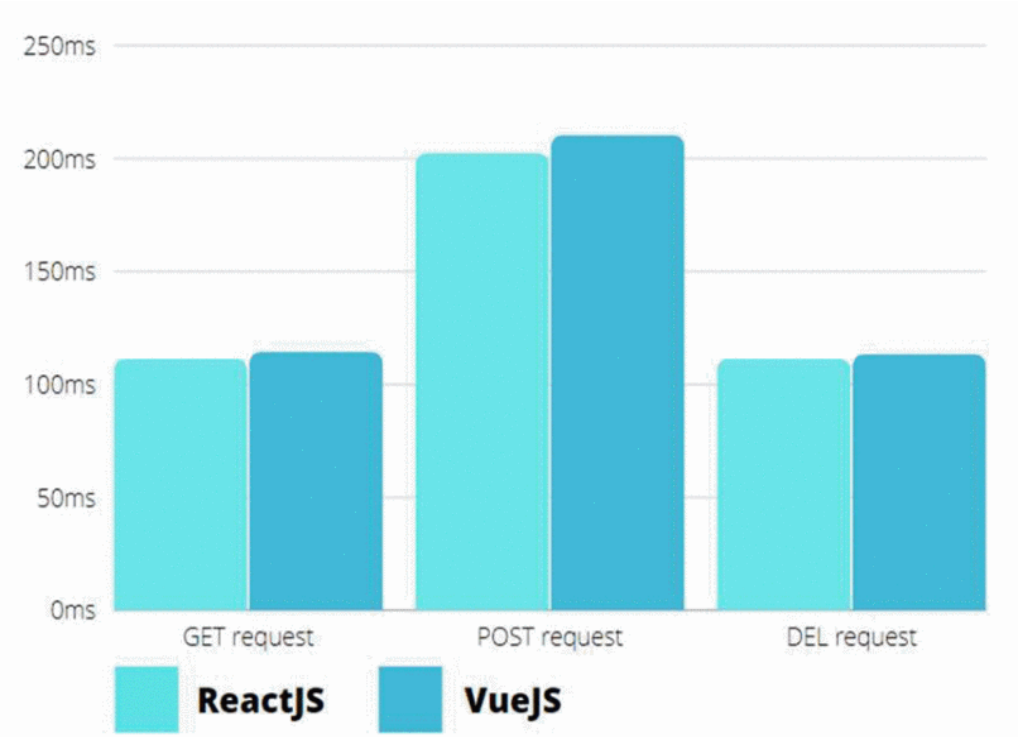
Sovelluksen osien manipuloiminen ohjelmointirajapinnoista noudetun datan perusteella on AJAX-tekniikkaa käyttävälle verkkosovellukselle yleinen tehtävä. Sen vuoksi sovelluskehysten suoriutumista on syytä tarkastella. Taulukossa 4 esitellään sovelluskehysten renderöintiaikoja erilaisia API-pyyntöjä toteuttaville komponenteille. React oli keskimäärin 3,7 ms nopeampi renderöimään GET-pyyntöjä tai uudelleenrenderöimään komponentteja, joiden tila muuttui API-pyynnön seurauksena. Molemmat kehykset myös tarvitsevat hieman enemmän aikaa renderöidessään POST-pyyntöä toteuttavan komponentin.

Taulukko 4 API-pyyntöä käyttävän komponentin renderöintiaika (Novac et al., 2021)



Taulukossa 5 esitellään aika, joka sovelluksilla kului useiden API-pyynnön noutamiseen. React oli jälleen hieman nopeampi verrattuna Vueen. Ero sovelluskehysten välillä on kuitenkin hyvin pieni. (Novac et al., 2021)

Taulukko 5 API-pyynnön toteuttamiseen kulunut aika



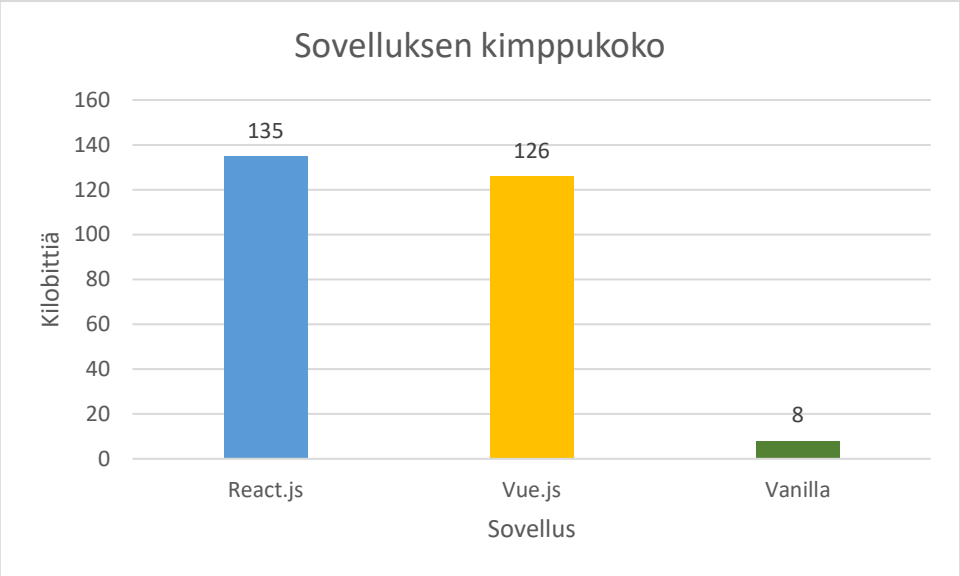
Tutkimuksessa todetaan Reactin olleen näillä mittareilla hieman nopeampi, mutta molemmat sovelluskehykset toimivat nopeasti ja kilpailukykyisesti. Tutkimuksessa myös huomautetaan ulkoisten tekijöiden, kuten API:n nopeuden, voivan olla merkittävä kokeessa mitattuihin aikoihin vaikuttava tekijä.

4.3 Sovelluksen kimppukoko

Sovelluksen kimpulla (eng. Application Bundle) tarkoitetaan sovelluksen verkkoa varten optimoidun tuotantoversion (eng. production build) kokoa, joka koostetaan automaattisesti sovelluskehyksen tuotantotyökalujen avulla. Kimppu koostuu HTML-tiedostosta ja koko ohjelman JavaScript-koodin minifioidusta versiosta. Pienemmän kimppukoon sovellus latautuu nopeammin ja sisältää vähemmän elementtejä selaimen analysoitavaksi (Figueiredoy et al., 2022).

Taulukossa 6 esitellään sovellusten kimppukoot. Tutkimuksessa havaittiin React-sovelluksen kimpun koon olleen 135,168 tavua, Vuen 126,976 tavua.

Taulukko 6 Sovellusten kimppukoot (Mukautettu Figueiredoy et al., 2022)

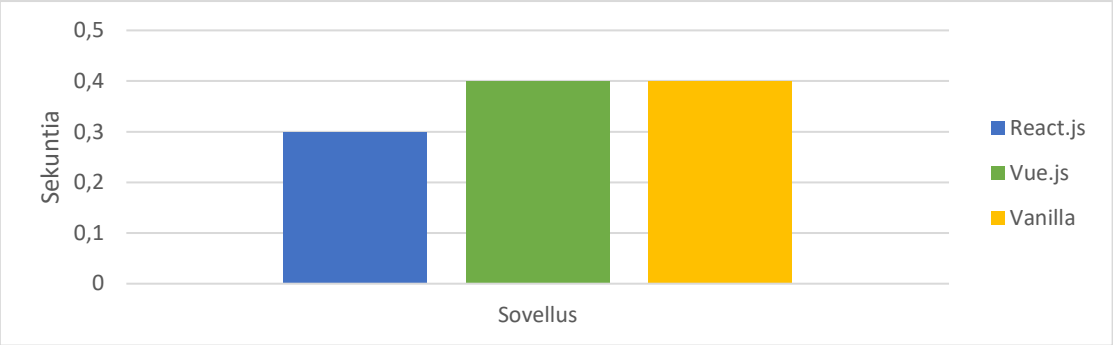


Sovellusten kokoerot eivät olleet merkittäviä, paitsi tilanteessa jossa sovelluksen koolla olisi erityistä painoarvoa. Tulosten perusteella tällöin järkevintä saattaisi kuitenkin olla puhtaan JavaScript-sovelluksen toteuttaminen.

4.4 Aika interaktiivisuuteen

Aika interaktiivisuuteen (engl. time to interactive) on sovelluksen reaktiivisuutta tarkasteleva mittari, joka mittaa kuinka pitkään ohjelmalla kestää palata täyteen inaktiiviseen valmiuteen vastaanottaa käyttäjältä uusia komentoja yli 5 sekuntia kestäneen tehtävän suorittamisen jälkeen (Chrome Developers, 2021; Mozilla, 2023b). Mittarin tarkastelu on hyvän käyttäjäkokemuksen kehittämisen kannalta tärkeää, sillä visuaalisesti valmiilta näyttävä sivu, joka ei kuitenkaan vastaanota käyttäjän komentoja, johtaa nopeasti käyttäjän turhautumiseen (Chrome Developers, 2021). Taulukko 7. esittelee tutkimuksessa havaitut ajat interaktiivisuuteen.

Taulukko 7 Aika interaktiivisuuteen (Muokattu (Figueiredoy et al., 2022)



Tutkimuksessa React-sovelluksen aika oli koko tarkastelun nopein, 0,3 sekuntia ja Vuen tutkimuksen keskitasolla, 0,4 sekuntia (Figueiredoy ym., 2022). Vaikka React-sovellus oli nopeampi kuin muut sovellukset, ei ero ollut merkittävä. Käyttäjäkokemuksen kannalta hyvä aika interaktiivisuuteen keskiverrolla mobiililaitteella on alle 5 sekuntia (Walton, 2019).

5 Pohdinta

Sovellusten teknisen suorituskyvyn näkökulmasta niin React kuin Vue luovat lopulta hyvin samanlaisia, kevyitä ja reaktiivisia sovelluskokonaisuuksia. Molemmissa tutkielmassa tarkastelluista tutkimuksissa havaittiin joitain eroja kehysten välillä, mutta lopulta todettiin niiden olevan hyvin pieniä.

Sovellusten kimppujen koot eivät juurikaan eronneet toisistaan. Ero kokojen välillä ei ollut huomattava, ja niiden keveyttä voidaan pitää seurauksena molempien sovelluskehysten keveyttä painottavasta suunnittelufilosofiasta (Figueiredoy et al., 2022). Pienemmän kokonsa takia Vue voisi kuitenkin soveltua käyttötapauksiin, joissa latausnopeudella on erityistä painoarvoa. Puhdas JavaScript on kuitenkin selvästi kooltaan pienin, joten jos sovelluskehystä ei tarvita kehitysprosessin tehostamiseen, se voisi tässä tapauksessa olla paras vaihtoehto.

DOM-manipulaationopeutta tarkastelevassa kokeessa mielenkiintoisia havaintoja oli kaksi: ensinnäkin Vue säilyi suhteellisesti yhtä tehokkaana kaikissa skenaarioissa, ja toiseksi Reactin Create-funktion hidastui selkeästi elementtien määrän kasvaessa. Vuen tasainen suoriutuminen voisi tehdä siitä sopivan niin pieniin kuin suuriin sovelluksiin, ja myös sovelluksiin, jotka mahdollisesti laajenevat tulevaisuudessa. Tutkimuksessa ehdotettiinkin Vuen mahdollisesti soveltuvan monipuoliseen yleiskäyttöön ja Reactin pienempien, nopeaa reaktiivisuutta tavoittelevien sovellusten toteuttamiseen (Figueiredoy et al., 2022). Aikaa interaktiivisuuteen -mittari antaisi myös Reactille pientä etua verrattuna Vueen ja myös puhtaaseen JavaScriptiin, mikä tukisi tätä ehdotusta.

Tehokkuusvertailua varten luodut sovellukset ovat usein pieniä ja hyvin yksinkertaisia. Kokeet ei välttämättä tuo esille niitä eroja, jotka nousevat esille ohjelman monimutkaisuuden kasvaessa. Suurempien testisovellusten toteuttaminen taas ei välttämättä ole tutkimuksen kannalta järkevää, sillä identtisten ominaisuuksien toteuttaminen ei välttämättä ole enää mahdollista. Jatkotutkimus suurempien sovelluskokonaisuuksien teknisestä suoriutumisesta voisi kuitenkin olla mielekästä yrittää toteuttaa, jotta sovelluskehysten kyvykkyyttä voitaisiin vertailla suurempien sovellusten kontekstissa. Reactin Create-funktion hidastuminen suurilla elementtimäärillä voisi viitata siihen, että erot nousisivat esiin juuri suurten sovellusten tehokkuudessa.

Kehyksillä tuotettujen sovellusten teknisen samankaltaisuuden vuoksi kehyksen valintaa voisi olla järkevämpää tarkastella muista näkökulmista. Reactin keskittyminen View-tason kehittämiseen tarkoittaa, että kehittäjän on valittava sovellukseensa myös muita frontendin ominaisuuksia toteuttavia kirjastoja. Sovelluksen riippuvuuksien lisääntyminen on haaste, joka olisi hyvä ottaa huomioon etenkin, jos kehitetyn sovelluksen tulisi olla pitkäikäinen.

Kokenut ohjelmoija todennäköisesti osaa valita React-projektin vaatimuksiin parhaiten sopivat lisäkirjastot. Aloittelevalle ohjelmoijalle tämä taas voi olla haaste, ja Vue voisi tarjota kaikki tarvittavat ominaisuudet sisältävän selkeän ja kokonaisvaltaisen pakettiratkaisun. Aloittelevalle kehittäjälle yhden kokonaisuuden hahmottaminen ja opettelu saattaisi olla helpompaa kuin usean kirjaston samanaikainen käyttäminen.

React on erittäin suosittu ja käyttäjiä on paljon, minkä vuoksi tukea oppimiseen on helppo löytää verkosta. Runsas ohjelmointipulmiin liittyvä keskustelu ja kattava tutoriaalitarjonta tekevät kirjaston käytön opiskelusta helppoa. Vuen suosio on kasvustaan huolimatta edelleen Reactiin verrattuna pientä (Stack Overflow, 2022). Vahva kehittäjäkulttuuri antaa mahdollisuuksia saada vastauksia kysymyksiin muilta kehittäjiltä, eikä samojen ongelmien ratkomiseen tarvitse käyttää liikaa aikaa.

6 Yhteenveto

Tutkielman tavoitteena oli avata näkökulmia sovelluskehyksien käyttämiseen yksisivuisten verkkosovellusten kehityksessä ja auttaa omiin tarpeisiin sopivan kehyksen valinnassa. Tarkasteluun valittiin suosituin sovelluskehys React.js sekä suosiotaan kasvattanut ja kehittäjien keskuudessa pidetty Vue.js.

Tutkielma esitteli yksisivuisen verkkosovelluksen toimintaperiaatteen ja edut verrattuna monisivuisiin sovelluksiin. Tämän jälkeen tutkielmassa esiteltiin Reactin ja Vuen taustoja, ja käytiin läpi kehysten ydinominaisuuksia ja haasteita. Tutkielmassa tarkasteltiin sovellusten tehokkuutta Figueiredoyn ja Novacin tekemien vertailututkimusten avulla, joissa vertailtiin kehyksillä kehitettyjen testisovellusten tehokkuutta erilaisilla mittareilla. Molemmilla sovelluskehyksillä kehitetyt sovellukset olivat kevyitä ja reaktiivisia, eikä niiden välillä ei havaittu suuria eroja tehokkuudessa.

Teknistä suorituskykyä vertailleiden tutkimusten perusteella tutkielma esittää, että sovelluskehyksillä kehitetyt sovellukset ovat yleisesti suorituskyvyltään hyvin samankaltaisia, eikä teknisen suorituskyvyn näkökulma ole välttämättä kaikkein oleellisin kehystä valitessa. Tutkielma kehottaa sovelluskehyksen valinnassa tarkastelemaan myös muita, kehittäjälähtöisempiä näkökulmia, kuten sovelluskehyksen kehittäjäyhteisöä ja projektiin osallistuvien kehittäjien kokemuspohjaa ja mieltymyksiä.

Lähdeluettelo

Borges, H., & Tulio Valente, M. (2018). What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform [Article]. *The Journal of Systems and Software*, *146*, 112–129. https://doi.org/10.1016/j.jss.2018.09.016

Chrome Developers. (2021, June 4). *Time to Interactive*. Google Developers. https://developer.chrome.com/docs/lighthouse/performance/interactive/

Edwin, N. M. (2014). Software Frameworks, Architectural and Design Patterns [Article]. *Journal of Software Engineering and Applications*, *7*(8), 670–678. https://doi.org/10.4236/jsea.2014.78061

Figueiredoy, C. C. L., De S.Russo, G., Bahiense-Junior, M. R. G., Mateus V. L, A., Dos Santos, L. M., Da Rocha, R. F., Bezerra, R. R., & Giuntini, F. T. (2022). Evaluating the performance of web rendering technologies based on JavaScript: Angular, React, and Vue. *The Institute of Electrical and Electronics Engineers, Inc. (IEEE) Conference Proceedings*. https://doi.org/10.1109/CLEI56649.2022.9959901

Greif, S., & Burel, E. (2022). *The State of JS 2022*. https://2022.stateofjs.com/en-US/

Hámori, F. (2022). *The History of React.js on a Timeline*.

Janssen, C. (n.d.). *Software Framework*. Retrieved March 26, 2023, from http://www.techopedia.com/definition/14384/software-framework

Kumar, T. (2024). *Fluent React* (Early Rele). O'Reilly Media, Inc.

Mozilla. (2023a, February 22). *SPA (Single-page application)*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Glossary/SPA

Mozilla. (2023b, February 22). *Time to Interactive*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Glossary/Time_to_interactive

Novac, C. M., Novac, O. C., Sferle, R. M., Gordan, M. I., Bujdoso, G., & Dindelegan, C. M. (2021). Comparative study of some applications made in the Vue.js and React.js frameworks. *2021 16th International Conference on Engineering of Modern Electric Systems, EMES 2021 - Proceedings*. https://doi.org/10.1109/EMES52337.2021.9484149

React.js. (2023). *React.js - A JavaScript library for building user interfaces*. https://legacy.reactjs.org/

Scott, E. A. (2016). *SPA design and architecture : understanding single page web applications* (1st edition) [Book]. Manning.

Smith, S. (2022). *Architechting Modern Web Applications with ASP.NET Core and Microsoft Azure* (M. Wenzel (ed.); v6.0). Microsoft Developer Division, .NET, and Visual Studio product teams. https://raw.githubusercontent.com/dotnet-

architecture/eBooks/main/current/architecting-modern-web-apps-azure/Architecting-Modern-Web-Applications-with-ASP.NET-Core-and-Azure.pdf

Stack Overflow. (2022). *Stack Overflow Developer Survey 2022*. https://survey.stackoverflow.co/2022/

Vuejs.org. (2019). *Comparison with Other Frameworks*. https://vuejs.org/v2/guide/comparison.html

Vuejs.org. (2023). *Vue.js - The Progressive JavaScript Framework*. https://vuejs.org/

Walton, P. (2019). *Time to Interactive (TTI)*. Web.Dev. https://web.dev/tti/#what-is-a-good-tti-score

You, E. (2016). *The State of Vue*. Medium. https://medium.com/the-vue-point/the-state-of-vue-1655e10a340a