TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

HENRI KUOKKANEN

RESPONSIIVISEN WEB-KÄYTTÖLIITTYMÄN TOTEUTUS TUOTANNONOHJAUSJÄRJESTELMÄÄN

Diplomityö

Tarkastaja: professori Tommi Mikkonen
Tarkastaja ja aihe hyväksytty
31. maaliskuuta 2017

TIIVISTELMÄ

HENRI KUOKKANEN: Responsiivisen web-käyttöliittymän toteutus tuotannonohjausjärjestelmään
Tampereen teknillinen yliopisto
Diplomityö, 44 sivua
Toukokuu 2017
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Pervasive Systems
Tarkastaja: professori Tommi Mikkonen

Avainsanat: Yhden sivun sovellus, responsiivisuus, JavaScript, React

Web-sovellukset ovat kasvavissa määrin korvaamassa natiivisovelluksia. Web-sovellukset ovat alustasta riippumattomia, jolloin yksi sovellus toimii useilla laitteilla. Modernit responsiiviset web-sovellukset ovat korvanneet vanhemmat staattiset web-sovellukset. Responsiiviset web-sovellukset hyödyntävät yleensä valmiita JavaScript-kirjastoja, jotka mahdollistavat nopean sovelluskehityksen sekä monipuoliset ominaisuudet web-sovelluksissa. JavaScript-kirjastoja on viime vuosina julkaistu useita, ja sopivan kirjaston löytäminen voikin olla yrityksille vaikeaa.

Tämän tutkimuksen tavoitteena oli löytää SW-Development yritykselle sopiva JavaScript-käyttöliittymäkirjasto. Yrityksen tuotannonohjausjärjestelmää ollaan kehittämässä web-teknologioihin perustuvaksi. Tarkoituksena on löytää yrityksen käyttöön sopiva JavaScript-käyttöliittymäkirjasto, joka sopii yrityksen web-pohjaiseen SWD^{PES} -sovellukseen. Valitun JavaScript-kirjaston pitää soveltua responsiivisen käyttöliittymän luomiseen. Lisäksi kirjastolla pitää pystyä tekemään single-page application (SPA) -sovelluksia. SPA-sovellukset ovat responsiivisia web-sovelluksia, joiden ei tarvitse ladata web-sivua uudestaan, kun käyttöliittymää päivitetään.

Diplomityön tutkimusmenetelmä oli kvalitatiivinen. Tutkimuksessa valittiin kriteerit, joihin useita JavaScript-kirjastoja verrattiin. Kriteereinä käytettiin esimerkiksi suorituskykyä ja helppokäyttöisyyttä. Tämän jälkeen kriteerien perusteella valittiin tekniikat, joita käyttäen toteutettiin responsiivinen demosovellus. Tälle demosovellukselle valittiin ennen toteutusta kriteerit, joiden perusteella arvioitiin demossa käytettyjen tekniikoiden soveltuvuutta yrityksen web-sovelluksiin.

Tutkimuksessa löydettiin sopivimmaksi kirjastoksi React. Reactia käytettiin demosovellusessa, jonka jälkeen sitä verrattiin demosovelluksen arviointikriteereihin. Tulosten perusteella React sopii yrityksen tuotannonohjausjärjestelmään, sillä se mahdollistaa responsiiviset web-käyttöliittymät, hyvän suorituskyvyn ja on helppokäyttöinen.

ABSTRACT

HENRI KUOKKANEN: Implementation of responsive web user interface for a production planning system
Tampere University of Technology
Master of Science Thesis, 44 pages
Toukokuu 2017
Master's Degree Programme in Information Technology
Major: Pervasive Systems
Examiner: Professor Tommi Mikkonen

Keywords: Single-page application, responsive, JavaScript, React

Web applications are increasingly replacing native applications. Web applications are independent of the platform, and therefore a single application runs on different platforms. Modern responsive web applications have been replacing older static web applications. Responsive web applications often utilize JavaScript libraries. Ready-made JavaScript libraries allow rapid application development, as well as versatile features in web applications. During the last few years multiple JavaScript libraries has been published. Finding the right library for the company, however, can be challenging.

The aim of this study was to find a suitable JavaScript user interface library for SW-Development. The company is developing a web-based production planning system software. The aim was to find a JavaScript-library which can be utilized with the web-based production planning system software SWD^{PES}. The library should be suitable for responsive web applications. Moreover, the library should be suitable for single-page applications (SPA). SPA's are responsive web applications which does not require web page reload when updating the user interface.

In this study, qualitative research was conducted. The study included popular JavaScript libraries, which were evaluated according to selected criterion. Criterions included for example performance and ease of use. After the evaluation of the libraries, a responsive demo application was developed. In addition, criteria for evaluation of the demo application was selected and the application was evaluated according to those criteria.

In this research, React was found the most suitable JavaScript library for the company. React was applied in demo application, which was evaluated with selected criterion. According to the results, React library is suitable for company's production planning system software, enabling responsive web applications, good performance and straightforward usability.

ALKUSANAT

Tämä diplomityö tehtiin ohjelmistoyritys SW-Development Oy:lle. Haluan kiittää SW-Developmentia ja kollegoitani sekä diplomityön aiheesta että avusta koko kirjoitusprosessin aikana.

Haluan myös kiittää ohjaajaani professori Tommi Mikkosta hänen loistavasta ohjauksesta ja avusta diplomityön tekemisessä. Kiitos myös muille läheisilleni ja ystävilleni, jotka ovat olleet tukena opiskelu-urani aikana.

Tampereella, 18.4.2017

Henri Kuokkanen

SISÄLLYSLUETTELO

1. JOHDANTO

Perinteisesti webin käytöstä suurin osa on tapahtunut työpöytätietokoneilla, ja usein käyttöjärjestelmänä on ollut Microsoft Windows [1]. Teknologinen kehitys viime vuosina on kuitenkin lisännyt mobiililaitteiden käyttöä, ja webin käyttäjät ovat siirtyneet suurissa määrin käyttämään mobiililaitteiden selaimia. Vuonna 2016 mobiililaitteilla käytettiin maailmanlaajuisesti ensimmäistä kertaa enemmän internetiä kuin työpöytätietokoneilla [1]. Lisäksi käyttöjärjestelmiin on tullut lisää vaihtoehtoja, kuten mobiililaitteiden iOS ja Android -käyttöjärjestelmät. Eri alustojen määrän kasvaessa sovelluksien tekeminen vain yhdelle alustalle ei ole kannattavaa, vaan sovelluskehittäjien pitäisi tukea useita eri käyttöjärjestelmiä.

Nykyaikaiset web-teknologiat, kuten AJAX ja HTML5, mahdollistavat monipuoliset responsiiviset käyttöliittymät ja sovellukset, joita voidaan suorittaa web-selaimella. Palvelimelta ladattava ja web-selaimessa suoritettava sovellus ei ole riippuvainen käyttöjärjestelmästä. Sovellusta ei myöskään tarvitse erikseen asentaa laitteelle ja sen päivittäminen onnistuu suoraan palvelimelta.

Diplomityö toteutettiin SW-Development-yritykselle. Diplomityössä tutkitaan, miten web-teknologioilla saadaan tehtyä responsiivinen käyttöliittymä MES-tuotannonohjausjärjestelmään. Tarkoituksena on selvittää, mitkä web-teknologiat sopivat SW-Developmentin nykyisiin ja tulevaisuuden web-sovelluksiin. Useat yrityksen tuotannonohjausjärjestelmän aikaisemmat versiot ovat perustuneet Windows-sovelluksiin. Tulevaisuudessa on tarkoitus siirtyä web-selaimilla käytettäviin ratkaisuihin. Teknisenä toiveena yrityksellä on saada tarkempaa tietoa responsiivisten web-käyttöliittymien ohjelmointiin sopivista JavaScript-kirjastoista. Kyseisiä käyttöliittymäkirjastoja on viime vuosina kehitetty useita. Kirjastojen vertailu voikin olla hankalaa.

JavaScript-kirjastot mahdollistavat web-sovelluksien kehittämisen paljon lyhyemmässä ajassa verrattuna siihen, että kehittäjä koodaisi kaiken alusta asti. JavaScript-kirjastoissa on paljon eroja. Yleensä ne sopivat responsiivisten web-sovellusten tekoon ja ne luovat web-sovelluksista SPA-sovelluksia. SPA-sovellukset ovat responsiivisia web-sovelluksia, joiden ei tarvitse ladata web-sivua uudestaan, kun käyttöliittymää päivitetään. Tällöin web-sovellus näyttää hyvältä sekä isommilla tietokoneen näytöillä että mobiililaitteiden pienillä näytöillä.

JavaScript-kirjastoista osa tarjoaa kokonaisen rungon responsiivisten SPA-sovelluksien tuottamiseen, kun osa sisältää vain pelkkiä käyttöliittymäelementtejä ja CSS-kirjastoja.

Lisäksi kirjastot ovat usein innovatiivisia ja ne mahdollistavat monia ratkaisuja, jotka eivät olisi mahdollisia ilman näitä kirjastoja. Ne auttavat myös sovellusten ylläpidossa ja parantavat niiden tietoturvaa, koska kirjastoja päivitetään usein. Usein ne kuitenkin vaativat päätelaitteelta uusimpia web-selaimia, joiden uusia rajapintoja ne hyödyntävät.

Tämä diplomityö koostuu seitsemästä luvusta. Luvussa 2 esitellään websovellusten perusteknologiat ja SPA-sovellukset. Luvussa 3 käydään läpi responsiivisten web-sovellusten keskeiset ominaisuudet. Luvussa 4 käsitellään SW-Developmentin tuotannonohjausjärjestelmää sekä sen kehitystä tulevaisuudessa. Seuraavaksi luvussa 5 tutkitaan ja vertaillaan web-teknologioita, sekä valitaan tekniikat demosovelluksen kehitykseen. Luvussa 6 esitellään tutkimuksessa toteutettu web-sovellus ja tutkimuksen tulokset. Viimeisenä luvussa 7 on yhteenveto.

2. SPA-SOVELLUKSET JA TEKNIIKAT

Tässä luvussa käsitellään diplomityöhön olennaisesti liittyviä käsitteitä ja tekniikoita, joilla voidaan rakentaa moderneja web-sovelluksia. Tekniikat pohjustavat työssä toteutettavaa demosovellusta ja sen ymmärtämistä. Luvussa esitellään myös yhden sivun sovellukset, joilla saadaan tehtyä dynaaminen käyttöliittymä web-sovellukseen. Yhden sivun sovelluksia käytetään nykyisin monissa internetpalveluissa. Yhden sivun sovellukset tarjoavat parhaimmillaan natiivisovelluksen tasolla olevan käyttökokemuksen ja ne ovat lähes kokonaan alustariippumattomia.

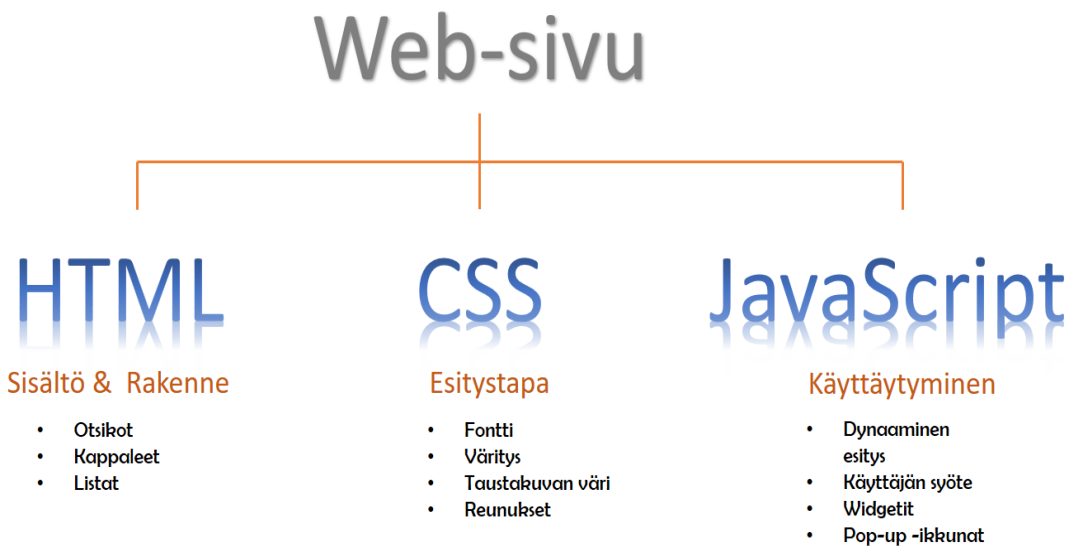
2.1 Web ohjelmointiympäristönä

Web-teknologioiden nopea kehitys viime vuosina on mahdollistanut monien palveluiden ja sovellusten siirtymisen internetiin. Aikaisemmin webin sivustot olivat hyvin yksinkertaisia ja esittivät lähinnä staattisia dokumentteja. Tällöin selaimesta tuli palvelimelle Hypertext Transfer Protocol (http) -pyyntö, minkä jälkeen palvelin palautti Hypertext Markup Language (HTML) -sivun selaimelle. Arkkitehtuuri oli tällöin asiakas-palvelin-mallinen (engl. client-server model). Tässä mallissa kaikki työ tapahtui palvelimella, jonka jälkeen web-selain esitti ainoastaan valmiin tuloksen. Nämä staattiset web-sivut eivät mahdollistaneet luontevaa interaktiota käyttäjän suuntaan, sillä sivu piti ladata kokonaan uudestaan, kun käyttäjä esimerkiksi navigoi uuteen näkymään. [2]

Modernit dynaamiset yhden sivun sovellukset (engl. Single-Page Application, SPA) ovat kehittyneet viime vuosina korvaamaan staattisia web-sivustoja. Ne mahdollistavat monimutkaisten interaktiivisten käyttöliittymien toteuttamisen web-selaimilla. Uudet versiot HTML ja Cascading Style Sheets (CSS) -standardeista, sekä nopeasti kehittyvät JavaScript-ratkaisut ovat SPA-sovellusten perusta. JavaScriptillä voidaan web-sivulla hallita muun muassa käyttäjän tekemää syötettä (engl. input) ja luoda monipuolisia animoituja käyttöliittymäelementtejä. Lisäksi SPA-sovelluksista voidaan tehdä responsiivisia, mikä mahdollistaa web-sovelluksen sopeutumisen erikokoisille näytöille ja laitteille. Aiemmin käytettiin usein erillisiä työpöytä- ja mobiilisivustoja kyseiseen tarkoitukseen. Responsiivisillä web-sivuilla voidaan luopua erillisistä mobiilisivustoita, mikä vähentää ylläpitäjien työtä, koska tällöin ei tarvitse ylläpitää kahta erillistä sivustoa. Toisin sanoen samaa informaatiota ei tarvitse kirjoittaa kahteen paikkaan. [2]

2.2 Tekniikat

Tässä kohdassa käsitellään tärkeimpiä tekniikoita responsiivisten web-sovellusten toteuttamisessa. HTML, CSS ja JavaScript ovat kolme internetstandardia, jotka määrittelevät yleisesti web-sivustojen toteutuksen (kuva 1). Näitä kolmea tekniikkaa käytetään käytännössä kaikissa web-sovelluksissa.



Kuva 1. *Web-sivuston perustekniikat [8].*

2.2.1 HTML5

Hypertext Markup Language 5 (HTML5) on kirjoitushetkellä HTML-standardin uusin versio, joka julkaistiin lokakuussa vuonna 2014. HTML ja sen eri versiot ovat olleet aina World Wide Web (WWW) -sivustojen merkintäkieliä. Sen alkuperäisenä tarkoituksena oli soveltua ohjelmointikieleksi tieteellisten dokumenttien semanttiseen kuvailemiseen. Uudemmat versiot ja muutokset mahdollistavat kuitenkin monien erityyppisten dokumenttien kuvailemisen. HTML-merkintäkielen vanhemmat standardit eivät sopineet hyvin modernien web-sovellusten käyttöön. Tähän ongelmaan ja muihin 2010-luvulla ilmenneisiin puutteisiin pyritään vastaamaan HTML5-standardin spesifikaatiossa. [3]

HMTL5-standardi sekä poisti että lisäsi useita elementtejä verrattuna aikaisempaan HTML4-spesifikaatioon. Poistettuja elementtejä olivat esimerkiksi <*center*> ja <*big*> -elementit, joilla voitiin muokata tekstin esitystapaa sivuilla. Uusi standardi kuitenkin määrittelee yleensä vaihtoehtoisen tavan esittää samat asiat kuin aikaisempi standardi. Myös edellä mainitut *big* ja *center* -määritteet pystytään korvaamaan CSS-tyylitiedoston muilla määritelmillä. HTML5:n uusista elementeistä hyödyllisimpiä ovat uudet multimedia (<*audio*>, <*video*>) ja grafiikkapiirros (<*svg*>, <*canvas*>) -elementit, joilla voidaan

korvata paljon käytettyjä multimedian selainlisäosia, kuten Adoben Flash Player tai Microsoftin Silverlight. [4]

2.2.2 JavaScript

JavaScript on korkean tason ohjelmointikieli, jota käytetään yleisesti web-sivustoilla HTML:n ja CSS:n kanssa. JavaScript esiteltiin jo vuonna 1995, ja siitä tuli ECMA-standardi (ECMA-262) vuonna 1997 [5]. JavaScript-kielestä käytetään nykyisin myös synonyymiä ECMAScript-nimitystä. Nimitys tulee ECMA-komitean standardeista, joista uusin on vuonna 2015 julkaistu ECMAScript 6.0 [6]. Kaikki nykyaikaiset web-selaimet, kuten Microsoft Edge, Mozilla Firefox sekä Google Chrome, tukevat JavaScriptiä. Selaimet osaavat käytännössä suorittaa web-sivustolla olevia JavaScript-koodeja (puhekielessä skripti), joilla voidaan lisätä web-sivun interaktiota käyttäjälle, hallita selaimen toimintaa sekä muokata selainikkunassa näkyviä dokumentteja [7]. JavaScriptin käyttö onkin yleisintä web-selaimissa, mutta sen käyttö ei rajoitu ainoastaan selaimiin, vaan sitä voidaan käyttää myös palvelinsovelluksissa. Se sopii myös muun tyyppisiin sovellutuksiin, kuten web-selaimien lisäosien koodaamiseen [7]. Tässä diplomityössä keskitytään kuitenkin web-selaimissa suoritettaviin JavaScriptiin ja SPA-sovelluksiin.

Modernit web-selaimet tukevat laajasti JavaScriptin uudempia versioita, mikä mahdollistaa yhdessä Document Object Modelin (DOM) kanssa dynaamiset HTML-sivustot (DHTML) [7]. Web-sivulla HTML-määrittelee sivun rakenteen, ja CSS-tyylitiedosto ulkoasun sekä muotoilun. Näiden lisäksi JavaScriptillä saadaan web-sivulle interaktiiviset elementit (kuva 1). Nykyaikaiset SPA-sovellukset käyttävät JavaScriptin tarjoamaa mahdollisuutta muokata DOMia dynaamisesti. Selaimet toimivat siis JavaScript-tulkkina, eli ne suorittavat palvelimelta ladatut skriptit itsenäisesti.

Ohjelmointikielenä JavaScript hyödyntää useita ohjelmointiparadigmoja ja sen syntaksi muistuttaa C ja Java -ohjelmointikieliä. Monet rakenteet näistä ohjelmointikielistä toimivat myös JavaScriptissä. JavaScript esimerkiksi sisältää *if*, *while*, ja *switch* -rakenteet. Yksi suurimmista eroista edellä mainittuihin on, että JavaScriptissä ei ole luokkia. Lisäksi funktiot JavaScriptissä ovat objekteja, jolloin ne voivat sisältää suoritettavaa ohjelmakoodia ja *property*-tietoja. JavaScriptin funktioita voidaan myös välittää kuten mitä tahansa muitakin objekteja [9].

Ohjelmassa 1 esitetään HTML-sivulla oleva JavaScript-skripti, joka lähettää käyttäjän *input*-kenttään kirjoittaman syötteen *myFunction*-funktiolle. Ohjelmasta 1 nähdään, miten JavaScript-koodi merkitään *script*-merkinnällä. Lisäksi ohjelmasta 1 selviää, miten kohdassa 2.3 esiteltävä *document*-objekti mahdollistaa oikean elementin valitsemisen HTML-sivulla.

```
<form id="form" action="form_action.asp">
  Input: <input type="text" name="fname"><br>
  <input type="button" onclick="myFunction()" value="Submit">
</form>

<script>
function myFunction() {
    document.getElementById("form").submit();
}
</script>
```

Ohjelma 1. *JavaScript-esimerkki tekstisyötteen poiminnasta.*

2.2.3 CSS

Cascading Style Sheets (CSS) on World Wide Web Consortium (W3C) -järjestön määrittelemä standardi, jonka avulla voidaan kuvailla HTML-dokumentin ulkoasua. CSS voi määritellä myös muiden dokumenttien, kuten Extensible Markup Language (XML) -dokumenttien ulkoasua, mutta tässä työssä käsitellään web-sivujen ulkoasun määrittelyä. W3C:n uusin standardi kielestä on CSS3. Se jakoi kielen moduuleihin ja esitteli uusia ominaisuuksia kieleen. CSS3 on taaksepäin yhteensopiva vanhempien CSS-standardien kanssa. W3C on esittänyt, että CSS3 standardin kehitys on jatkuvaa, ja uusia ominaisuuksia lisätään standardiin niiden valmistuessa. [10]

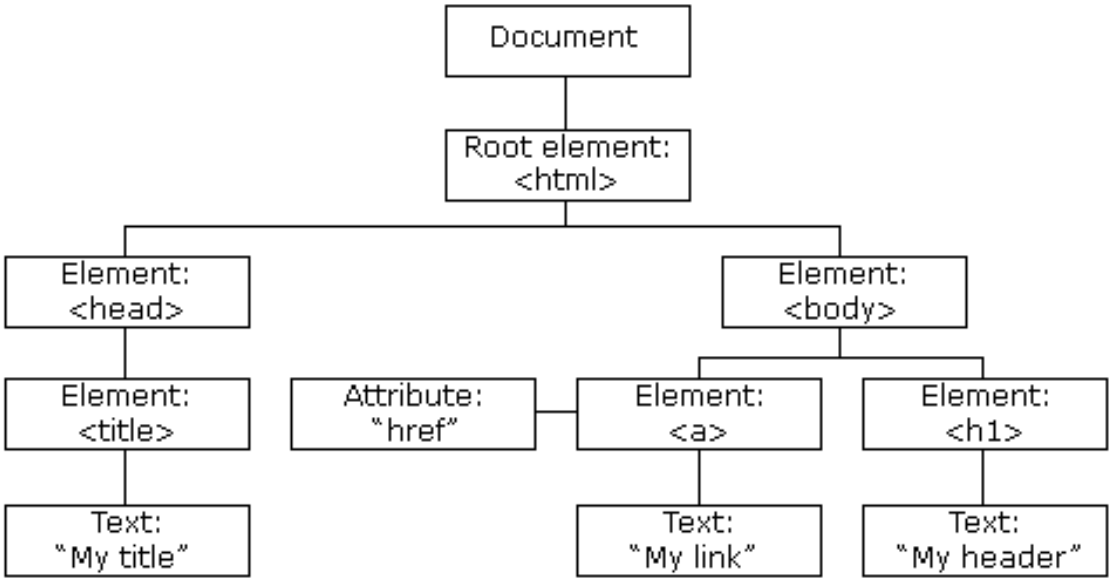
CSS-tyylitiedostot toimitetaan yleensä web-sivulla omana *.css*-tiedostopäätteisenä tiedostona. Se mahdollistaa tyylien, designin ja ulkoasun asettamisen kaikille HTML-dokumentin elementeille. Tällöin HTML-dokumentissa ei tarvitse säilyttää tietoa, miten elementit kuten otsikot tai kuvat pitäisi ruudulla esittää. Tyylitiedostoon voidaan sisällyttää myös vaihtoehdot esittämiseen erilaisille laitteilla. Tällöin sivusto osaa valita sopivan ulkoasun erikokoisille näytöille ja laitteille. Lisäksi samaa CSS-tyylitiedostoa voidaan käyttää monilla sivuilla. [10]

2.3 DOM

Document Object Model (DOM) on W3C:n määrittelemä standardi dokumenttien käsittelyyn, jota hyödynnetään web-sivustoilla. W3C:n määritelmässä DOM esitellään seuraavasti: DOM on alusta ja ohjelmointikielestä riippumaton rajapinta, joka mahdollistaa ohjelmien ja skriptien käsitellä sekä päivittää dynaamisesti sisältöä, rakennetta ja dokumenttien tyyliä [11]. W3C DOM määrittely on jaettu kolmeen osuuteen:

1. Core DOM - standardimalli kaikille dokumenttityypeille
2. XML DOM - standardimalli XML-tyyppisille dokumenteille
3. HTML DOM - standardimalli HTML-tyyppisille dokumenteille.

Kuvassa 2 on esimerkki puurakenteesta HTML DOMissa. DOM määrittelee loogisen puutyyppisen rakenteen dokumentille, sekä tavan valita osia dokumentista ja manipuloida niitä. Nämä yksittäiset elementit HTML-sivustolla esitetään HTML DOMissa objekteina. Näille elementeille DOM määrittelee ominaisuudet, metodit sekä *event*-tapahtumat. DOMin ja JavaScriptin yhdistelmällä pystytään luomaan dynaamisia HTML-sivuja, sekä SPA-sovelluksia. [11]



Kuva 2. *HTML DOMin puurakenteinen objektimalli [11].*

Kuvan 2 DOM-dokumentti esittää yksinkertaista web-sivua. JavaScriptillä voidaan käsitellä mitä tahansa elementtiä sivusta, mikä tapahtuu kutsumalla ensimmäisenä *document*-objektia. Tämän jälkeen voidaan hakea tarvittavat elementit *document*-funktioilla. Esimerkiksi kutsumalla *getElementById*-funktiota ja syöttämällä sille parametrina elementin *id*-tunnus, saadaan noudettua DOMista haluttu elementti. Jos haettu elementti on *html*-elementti, sitä voidaan muokata muun muassa *innerHTML*-metodin avulla (ohjelma 2).

```
1. <!DOCTYPE html>
2. <html>
3. <body>
4.
5. <p id="test"></p>
6.
7. <script>
8. document.getElementById("test").innerHTML = "Hello World!";
9. </script>
10.
11.</body>
12.</html>
```

Ohjelma 2. *DOM-esimerkki, jossa muutetaan HTML-elementin tekstiä.*

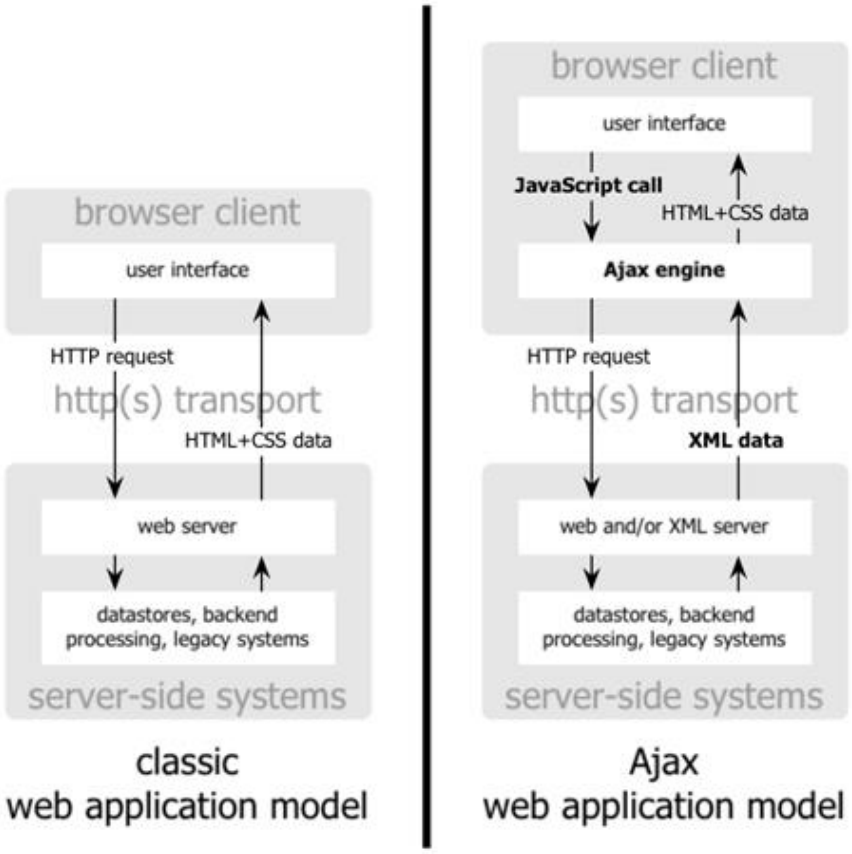
2.4 SPA

Single-page application (SPA) -sovelluksilla tarkoitetaan moderneja yhden sivun web-sovelluksia. Yhden sivun web-sovelluksissa sovellus ladataan yhden kerran web-selaimeen, jonka jälkeen ei tehdä uusia sivulatauksia silloin, kun käyttöliittymää päivitetään selaimessa. Käytännössä SPA-sovellus ladataan käyttäjän web-selaimeen, jonka jälkeen sovelluksen suoritus tapahtuu web-selaimessa. Tällöin palvelimen rooli web-sovelluksessa on pienempi kuin vanhemmissa staattisissa web-sovelluksissa. SPA-sovelluksilla saavutetaan natiivisovelluksia lähempänä oleva käyttöliittymä kuin tavallisilla web-sivustoilla. [12]

SPA-sovelluksissa käyttöliittymää päivitetään ja dataa siirretään palvelimelle, ilman tarvetta päivittää web-selaimen sivua. Kaikki tarvittava HTML, CSS ja JavaScript -data ladataan siis heti ensimmäisellä sivun latauskerralla. Käyttäjälle tämä näkyy siten, että sivu ei päivity uudestaan tai ohjaa uuteen osoitteeseen käyttäjän interaktion takia. Ensimmäisen latauksen jälkeenkin SPA-sovellus noutaa dataa palvelimelta, mutta tämä voidaan tehdä dynaamisesti taustalla. [12]

SPA-sovelluksien dynaaminen datan siirto palvelimen ja web-selaimen välillä tehdään AJAX-tekniikoilla. SPA-sovellukset alkoivat kehittyä nopeasti, kun AJAX-tekniikoita alettiin käyttää vuonna 2005. AJAX ei itsessään ole yksittäinen uusi tekniikka, vaan laaja kokoelma web-teknologioita, jotka mahdollistavat asynkroniset pyynnöt palvelimelle sekä yksittäisten osioiden uudelleen piirtämisen web-sivulla. Teknologiayhtiö Googlen julkaisemien Google Mail (julkaisu 2004) ja Google Maps (julkaisu 2005) -web-sovelluksien, sekä niissä käytetyt AJAX-tekniikoiden vauhdittivat tekniikoiden muuttumista W3C:n määrittämiksi standardeiksi. [13]

Kuvassa 3 esitellään tavallinen web-sovelluksen arkkitehtuuri sekä AJAX-tekniikoita käyttävän modernin web-sovelluksen arkkitehtuuri. Kuvasta 3 nähdään, että SPA-sovelluksessa käytetään AJAXia (AJAX engine), ja sovellus käyttää JavaScript-pyyntöjä (engl. JavaScript call) käyttöliittymän muokkaamiseen. SPA-sovelluksessa web-sivun dokumentit muokataan siis web-selaimessa ja palvelimelta tulee vain esimerkiksi XML-dokumentteja.



Kuva 3. *Tavallisen web-sovelluksen ja SPA-sovelluksen arkkitehtuurit [15].*

AJAXin asynkronisessa palvelinkommunikaatiossa käytetään selaimen XMLHttpRequest (XHR) -rajapintaa. XHR on W3C:n virallistama standardi, jota kaikki modernit selaimet tukevat. Se tarjoaa metodit datan siirtoon selaimen ja web-palvelimen välille. AJAX ja XHR -siirroissa käytetään yleensä XML tai JSON -dataa, mutta myös muita tiedostomuotoja on mahdollista käyttää. Käytännössä AJAXin avulla hoidetaan tiedonsiirto SPA-sovelluksesta palvelimelle, josta AJAXilla palautetaan HTML, CSS ja muu data käyttöliittymätasolle (kuva 3). [14]

3. RESPONSIIVINEN WEB

Responsiivinen web-suunnittelu (engl. Responsive Web Design, RWD) tarkoittaa www-sivustojen luomista siten, että sivustot sopeutuvat päätelaitteen näytön koon, resoluution, sekä muiden ominaisuuksien mukaisesti [16]. Termin esitti ensimmäisen kerran Ethan Marcotte vuonna 2010 [17]. RWD mahdollistaa web-sovellusten sopimisen ruudulle sekä tietokoneilla että mobiililaitteita käytettäessä ilman, että käyttäjän tarvitsee muuttaa sivun kokoa tai vierittää sivua horisontaalisesti. Responsiivisessa web-suunnittelussa hyödynnetään HTML ja CSS -tekniikoita sivuston koon muuttamiseksi näytölle sopivaksi. [18]

RWD:n suurin hyöty saadaan, kun pystytään luomaan yksi sivusto jokaiseen tarpeeseen, mikä helpottaa sivuston ylläpitoa. Sivuston ylläpitäjän ei tarvitse luoda uusia sivuja pelkästään layout-asettelua varten, vaan samalle sivulle pystytään lisäämään tarvittaessa puuttuvat layout-ohjeet. Nykyisin sivuja jaetaan esimerkiksi sosiaalisessa mediassa. RWD:n avulla saadaan yhtenäinen kokemus kaikille käyttäjälle, vaikka käyttäjien päätelaitteet eroaisivat toisistaan. Sivu voidaan esittää myös ilman horisontaalista vierityspalkkia, joka voi näkyä työpöytäkäyttäjän web-selaimessa, mikäli selain ei ole kokoruututilassa. [17]

3.1 RWD:n avainominaisuudet

Tässä kohdassa käydään läpi responsiivisen web-sovelluksen tärkeimmät suunnitteluperiaatteet. Kaikki responsiiviset web-sovellukset käyttävät seuraavana käsiteltäviä periaatteita.

Responsiivisten web-sivustojen suunnittelussa valitaan aluksi lähestymistavaksi ”mobiili-edellä” (engl. mobile first) tai ”työpöytä-edellä” (engl. desktop first). Tällä tarkoitetaan sitä, että sivusto suunnitellaan lähtökohtaisesti sopivaksi toiselle lähestymistavalle, jonka jälkeen mediakyselyiden avulla asetetaan sivusto mukautumaan erikokoisille näytöille ja resoluutioille. Kuvassa 4 havainnollistetaan resoluution vaikutusta responsiivisen web-sivun ulkoasuun. [17]

Kuva 4. *Elastinen ruudukkoasettelu eri resoluution näytöillä [19]*

Kuvassa 4 nähdään, miten kolmella erikokoisella näytöllä esitetään sama informaatio. Kuvan ensimmäisen näytön resoluutio on 1024 pikseliä, jolloin oranssilla värillä olevat elementit asetetaan vierekkäin ruudulle. Kahdessa muussa näytössä (768 ja 320 pikseliä) samat oranssilla esitetyt elementit asetetaan päällekkäin, jotta niiden informaatio saadaan esitettyä. Kuvan pienimmällä 320 pikselin näytöllä kaikki elementit on aseteltu päällekkäin, jolloin käyttäjälle ei näy horisontaalista vierityspalkkia. [19]

Keskeiset suunnitteluperiaatteet responsiivisten web-sivujen tekemiseen ovat [17]:

1. Mediakyselyt (engl. media queries), jotka kysyvät esimerkiksi päätelaitteen näytön resoluutiotietoja.
2. Liikkuva ruudukko -asettelu (engl. fluid grid layout). Eli sivusto käyttää joustavaa mittasuhteisiin perustuvaa asettelu-layoutia (kuva 4).
3. Joustavat kuvat, videot ja muu media. Lisäksi hyödynnetään dynaamista koon muuttamista (engl. resizing). Tämä tarkoittaa, että sivuston pitää pystyä automaattisesti muuttamaan asiakkaan päätelaitteen mukaan layout-asettelua, sekä kuvien ja muun median kokoa. Tarvittaessa pitää pystyä leikkaamaan (engl. crop) median kokoa, jos media ei sovi päätelaitteen ruudulle valitulla resoluutiolla.

3.2 Mediakyselyt

Mediakyselyt tarkoittavat CSS3-standardissa esiteltyä moduulia, joka määrittelee lausekkeet, joilla HTML-sivun esitystapaa voidaan muokata päätelaitteen ominaisuuksien mukaan [20]. Mediakyselyt ovat perustana RWD:n ominaisuudelle, jossa HTML-sivusto osaa mukautua päätelaitteelle. Mediakyselyn tyyppisiä ominaisuuksia oli ensimmäisen kerran jo CSS2-standardissa, mutta modernit mediakyselyt esiteltiin ensimmäiseksi

CSS3-standardissa. Ne sisältävät esimerkiksi erilaisia mediatyyppejä ja ylimääräisiä kriteereitä, kuten ruudun p. Ohjelmassa 3 esitetään yksinkertainen mediakysely, jossa määritetään taustaväriksi vihreä, kun ruudun koko on vähintään 480 pikseliä leveyssuunnassa.

```
1. @media screen and (min-width: 480px){
2. body {
3. background-color: lightgreen;
4.   }
5. }
```

Ohjelma 3. *Mediakysely, jossa tarkistetaan ruudun leveys pikseleissä.*

RWD:ssä web-sivusto kysyy CSS3-kyselyillä tietoja päätelaitteesta. Mediakyselyt ovat ainoa tapa saada tietoja päätelaitteen ominaisuuksista. Mediakyselyjen tietoja voidaan sen jälkeen hyödyntää web-sivun elementtien piirtämisessä näytölle. Mediakyselyt koostuvat median tyypistä sekä nollasta tai useammasta ehdollisesta lausekkeesta, jotka voidaan evaluoida todeksi tai epätodeksi (engl. true/false). Ohjelmassa 3 median tyyppi on *screen*, joka on näyttöruudun ominaisuus. Lisäksi ehdollisena lausekkeena tarkastetaan, jääkö ruudun pikselimääräinen leveys alle 481 pikseliä. Mediakyselyiden ehtolausekkeet voivat sisältää esimerkiksi *width*, *height* tai *resolution* -kyselyitä. Niiden avulla pystytään valitsemaan päätelaitteelle sopivat CSS-tyylitiedostot ja mediatiedostot. [20]

3.3 Elastinen ruudukkoasettelu

Responsiivisessa web-suunnittelussa tarkoituksena on saavuttaa erikokoisille näytöille sopiva web-sivun ulkoasu. Tähän tarkoitukseen käytetään elastista ruudukkoasettelua (engl. fluid-grid layout). Vanhastaan sivut suunniteltiin typografiseen ruudukkoon, joka koostui riveistä sekä kolumneista, joihin moduulit voitiin sijoittaa (kuva 5). Typografisessa ruudukossa komponentit sijoitettiin käyttäen pikseliyksiköitä. Siitä seurasi, että ruudulla esitettävien asioiden koko riippui näytön pikselitiheydestä sekä pikselien kokonaismäärästä selainikkunassa. [19]

Kuva 5. *Typografinen sijoittelu [19].*

Responsiivisessa web-suunnittelussa pyritään luopumaan moduulien sijoittelusta absoluuttisilla yksiköillä, kuten pikselimääräisillä (*px*) yksiköillä. Sen sijaan sijoittelussa otetaan huomioon konteksti käyttämällä mediakyselyjä, jonka jälkeen lasketaan esitettäville asioille suhteellinen koko. Suhteellinen koko selvitetään kaikille moduuleille, sekä niiden välissä oleville tiloille ja etäisyyksille. Laskennassa voidaan käyttää esimerkiksi fonttikoon määrittämiseen kaavaa:

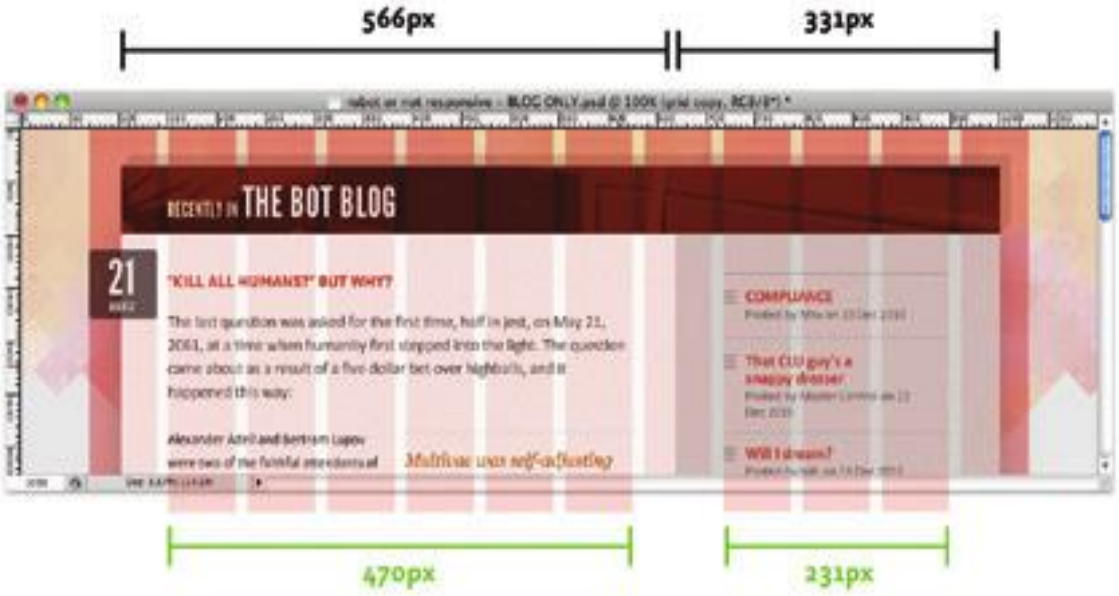
$$target \div context = result. \quad (1)$$

Kaavassa 1. *target* voi olla esimerkiksi fonttikoolle 24 pikseliä, kun sen ympäröivän elementin (*context*) koko on oletuksena 16 pikseliä. Tällöin kaavalla saadaan laskettua fonttikoolle suhteellinen tulos *1.5 em.* Tämä tarkoittaa, että fonttikokoa muutetaan aina 1.5-kertaiseksi sen *parent*-elementin fonttikokoon verrattaessa.

Yksikköinä, edellä mainitulle suhteellisen koon määrittämiselle, voidaan käyttää esimerkiksi prosenttiyksikköä (*%*), tai *em*-merkintää. Sekä *em*-yksikkö että prosenttiyksikkö kertovat kyseisen elementin koon suhteessa sitä ympäröivään elementtiin. [19]

Kuvassa 6 esitetään 12-sarakkeen levyinen ruudukkoasettelu. 12 saraketta on suurin sarakemäärä, jota CSS3-standardi tukee oletuksena, kun käytetään elastista ruudukkoasettelua. Kuvassa 6 esitetään kaksi leveyssuunnassa olevaa tekstilohkoa, joille voidaan laskea suhteellinen koko kaavalla: *470 px / 566 px = 83%.* Tekstin lisäksi kuvassa näkyy tekstialueen marginaalit, jotka ovat 566 pikseliä ja 331 pikseliä. Kun selainikkuna on tätä

suurempi, näytetään niiden ulkopuolella pelkkää taustakuvaa. Jos selainikkunaa pienennettäisiin sarakkeiden kokonaisleveyden alle, siirtyisivät tekstilohkot ruudulla päällekkäin. Näin estettäisiin horisontaalista vieritystä ruudulla. [19]



Kuva 6. *Esimerkki 12 sarakkeen levyisestä web-sivusta [19]*

3.4 Joustavat mediat

Joustavat mediat tarkoittavat kuvia, videoita ja muita mediaelementtejä, joita web-sivuilla käytetään [19]. RWD:ssä median koko pyritään säätämään sopivaksi eri näytöille, mikä voidaan toteuttaa asettamalla median suhteelliseksi kooksi (*max-width*) arvo *100 %:a*. Arvo voidaan määrittää CSS-tiedostossa medialle, jolloin se sovittaa median sopimaan ympäröivään *parent*-elementtiin. *Max-width* CSS-ominaisuus toimii kaikissa uudemmissa selainversioissa, mutta Internet Explorer ja Mozilla Firefox -selaimien vanhemmat versiot (Internet Explorer versio 6, Mozilla Firefox versio 2) eivät tue kyseistä ominaisuutta [21]. Internet Explorer version 6, sekä sitä aikaisemmat, voivat hyödyntää Microsoftin omaa *AlphaImageLoader*-funktiota, joka piirtää median vastaavalla tavalla kuin CSS3:n *max-width* arvo. Nykyisin valtaosa käytössä olevista web-selaimista osaa hyödyntää kyseisiä ominaisuuksia [22].

4. TUOTANNONOHJAUSJÄRJESTELMÄN NYKYINEN TILANNE JA TULEVAISUUS

Tässä luvussa käsitellään yleisellä tasolla SW-Development Oy:n (SWD) tuotannonohjausjärjestelmää, sekä sen tämänhetkistä tilaa ja tulevaisuuden kehityskohteita. Lisäksi luvussa esitellään MES-järjestelmien toimintaa ja niiden suhdetta muihin olemassa oleviin tuotannonohjauksen järjestelmiin.

SW-Development Oy on toimitusketjun tehokkuuden parantamiseen keskittynyt asiantuntijayritys. Yrityksen kehittämä Planning Efficiency System (SWD^{PES}) -tuotannonohjausjärjestelmä mahdollistaa asiakkaille tehokkaammat tuotanto- ja logistiikkaprosessit. Niiden saavuttamiseksi SWD^{PES} hyödyntää monia edistyneitä teknologioita kuten ajoitusmenetelmiä sekä heuristisia ja matemaattisia menetelmiä. [23]

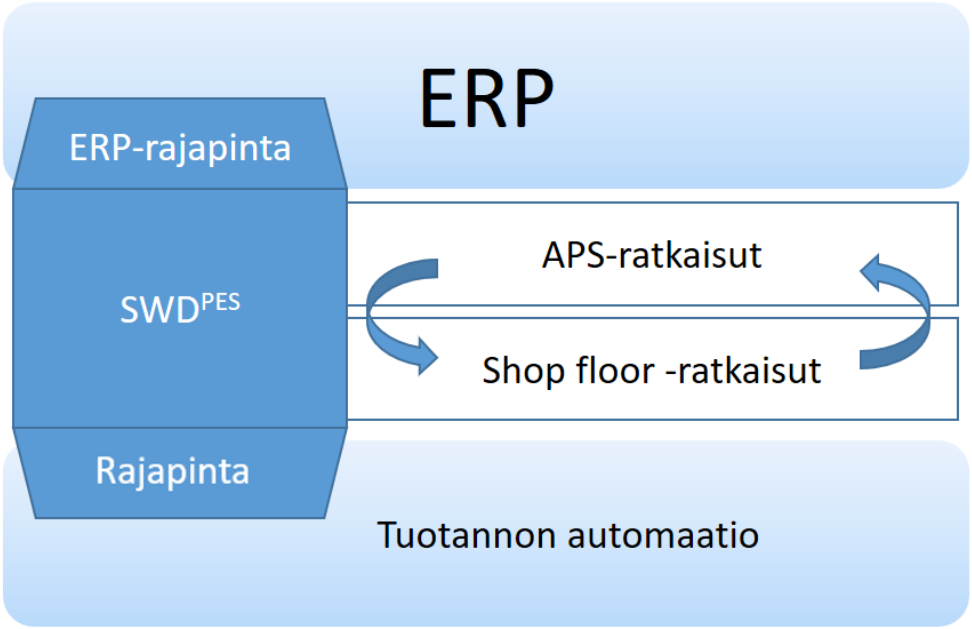
4.1 PES-järjestelmä tuotannonohjauksessa

SW-Development Oy:n Planning Efficiency System (SWD^{PES}) -ohjelmisto on kehitetty teollisuuden tuotannonsuunnittelun tehokkuuden lisäämiseen. SWD^{PES}-ohjelmistolla on mahdollista suunnitella toimitusketjun toimintaa ja tuotantoa, viikko- (karkeasuunnittelu), kuukausi- (karkeasuunnittelu) ja päivätasolla (hienosuunnittelu). Ohjelmistossa on hyvät valmiudet integroitua yritysten muihin tietojärjestelmiin, kuten paljon käytettyihin Enterprise Resource Planning (ERP) -järjestelmiin. Kyky tiedonvaihtoon usean erityyppisen järjestelmän kanssa onkin selkeä SWD^{PES}-järjestelmän etu suhteessa kilpaileviin järjestelmiin. Käytännössä ohjelmisto operoi ERP-järjestelmää tarkemmalla tasolla tehtaan tuotannossa. Se voi kuitenkin hyödyntää ERPin rajapintoja ja samoja tietokantoja. [23]

SWD^{PES} on luokiteltavissa Advanced Planning & Scheduling (APS) ja Manufacturing Execution systems (MES) -järjestelmiksi. APS-järjestelmät on suunniteltu paikkaamaan ERP-ohjelmistojen puutteita tuotannon hienosuunnittelussa [24]. MES-ohjelmistot yhdistävät tuotannonsuunnittelun liikkeenjohdon tasolla ja tuotannon (engl. shop floor) tason järjestelmät [25].

SWD^{PES} toimii yleensä ERP-toiminnanohjausjärjestelmien alapuolella tarjoten tarkempaa tietoa tuotannosta ja työkalut yksityiskohtaisempaan tuotannonsuunnitteluun (kuva 7). ERP toimii yleisesti karkealla tasolla ja se ei mahdollista nopeaa reagointia muuttuviin tilanteisiin tuotannossa [26]. SWD^{PES} -ohjelmistolla voidaan sen sijaan suunnitella toimitusketjun toimintaa sekä yrityksen tuotantoa päivä-, viikko- ja kuukausitasoilla. Kuvassa

7 esitetään PES-järjestelmä suhteessa muihin tuotannonohjausjärjestelmiin. Kuvasta 7 näkee kuinka SWD^{PES} yhdistää useiden järjestelmien palvelut yhdeksi kokonaisuudeksi.



Kuva 7. *SWD PES -järjestelmän suhde muihin tuotannonohjauksen järjestelmiin [27].*

Kuvassa 7 esitetyt shop floor control -ratkaisut (SFC) ovat ohjelmistoja, jotka tarjoavat metodit tuotantotilausten priorisointiin, jäljittämiseen ja raportointiin. Käytännössä tämä sisältää esimerkiksi proseduurit varastojen, työntekijöiden, kuorman sekä laitteiden kuormituksen seurantaan. Kyseiset proseduurit sisältävät tarkat tiedot eri tuotantotapahtumien sen hetkisistä tiloista, tarjoten tuotannonsuunnittelijoille työkalut tuotannon järjestämiseen. SFC:tä käytetäänkin yleensä ERP-ohjelmistojen jälkeen, kun tuotanto- ja ostotilaukset on jo suunniteltu. [28]

Automaatiotasolla kuvassa 7 tarkoitetaan yleisesti ISA-95 -standardissa esitettyjä 0, 1, 2-tasoja. ISA-95 on Yhdysvaltalaisen International Society of Automation (ISA) -järjestön vuosina 2000 – 2005 määrittelemä standardi tuotannonohjausjärjestelmien rakenteesta ja tehtävistä. Standardin määrittelemästä tasoista käytetään myös yleisesti nimitystä prosessienhallintataso. Käytännössä automaatiotaso sisältää varsinaista tuotantoa seuraavat automaatiojärjestelmät. Ne keräävät reaaliaikaista tietoa tuotannon etenemisestä.

Kuvan 7 rajapinta automaatiotasolle koostuu kaksisuuntaisesta yhteydestä SWD^{PES} -sovelluksen, sekä tuotannonhallinta- ja muuta dataa keräävien lattiatasonjärjestelmien välille. Käytännössä tämä tarkoittaa sitä, että tuotannonohjausjärjestelmä kerää tietoja sekä lattiatason laitteistoilta että työntekijöiltä prosessien tiloista ja laitteiden tapahtumista.

Myös erityyppiset hälytykset voidaan välittää kyseisen rajapinnan kautta. Automaatiotaso vastaavasti saa ylemmältä järjestelmältä tuotantosäännöt ja ohjeet. Nämä käskyt voivat sisältää työohjeita, tuotekohtaisia käskyjä, aikatauluja ja esimerkiksi eri operaatioihin tai materiaaleihin liittyviä turvallisuussääntöjä. [29]

SWD^{PES} sisältää myös tuotannonsuunnittelijoille monia työkaluja, joilla pystytään suunnittelemaan tuotantoa karkea- ja hienotasoilla. Järjestelmässä hienosuunnittelulla tarkoitetaan päiväkohtaista suunnittelua. Karkeasuunnittelu tarkoittaa viikko- tai kuukausitason suunnittelua. Karkeatasolla tuotannonsuunnittelijalle näytetään tuotannon tiedot vain viikoittain. Suunnittelijat voivat myös käyttää sekä luoda skenaarioita, joiden avulla suunnittelua on mahdollista tehdä hiekkalaatikkoympäristössä. Lisäksi tuotantoa on mahdollista optimoida optimointityökaluja käyttäen. Kuvassa 7 esitetyt shop floor -ratkaisut mahdollistavat työntekijöille tehtävät tuotantosuunnitelmat. Tällöin työntekijälle voidaan tarjota tehtävälistaus, joka sisältää tehtävien tarkat tiedot. [23]

Lisäksi SWD^{PES} tarjoaa raportointityökaluja ja -mittareita tehokkuuden selvittämiseen. Shop floor -tasolla mittarit esittävät, kuinka tehokkaasti valmistuskapasiteetti on käytössä. Ohjelmiston eri näkymät esittävät datan eri muodoissa, jolloin pystytään valitsemaan eri tilanteisiin parhaiten sopivat näkymät ja mittarit. SWD^{PES}:issä valitut tuotantosuunnitelmat ja muut ratkaisut on mahdollista yhdistää ERP:iin, sekä muihin mahdollisiin yrityksen automaatiojärjestelmiin. [23]

4.2 Tulevaisuuden tarpeet SWD^{PES} -järjestelmässä

SWD^{PES} järjestelmä toimii vuonna 2016 Windows-työpöytäsovelluksena ja tulevaisuuden kehitystoiveena yrityksellä on saada järjestelmä toimimaan useammalla alustalla, mukaan lukien käyttöliittymät mobiililaitteissa. Tällä hetkellä järjestelmää on käytetty työpöytäsovelluksen lisäksi myös tablet-laitteissa tietyissä tapauksissa. Tämän hetkisessä toteutuksessa on kuitenkin puutteena, että se ei mukaudu erityyppisiin päätelaitteisiin, esimerkiksi silloin kun päätelaitteen näytön resoluutio vaihtuu. Tästä voi seurata käytettävyysongelmia, joita on vaikea testata erilaisilla päätelaitteilla. [23]

Yrityksellä on kehityksessä web-selaimella käytettävä versio SWD^{PES}-järjestelmästä, joka on räätälöitävissä asiakkaan tarpeiden mukaan. Tarkoituksena on luoda responsiivinen web-selaimella käytettävä sovellus, joka toimii erikokoisilla näytöillä skaalautuen käyttäjälle loogisesti. Tällöin sovellus ei ole riippuvainen laitevalinnoista ja sitä voidaan käyttää myös kosketusnäytöillä ilman erillistä hiirtä. [23]

4.3 SWD^{PES} -järjestelmän nykyinen tekninen toteutus

SWD^{PES} -ohjelmisto toimii Microsoft Windows -käyttöjärjestelmällä varustetuissa tietokoneissa. SWD^{PES} on toteutettu C#-ohjelmointikieltä ja .Net -teknologia-alustaa hyödyntäen. Tietokoneelle pitääkin olla asennettuna .Net Framework -ohjelmistokehys, jotta sovellusta voidaan käyttää. C#-ohjelmointikieli on kehitetty Microsoftin toimesta ja sen ensimmäinen versio julkaistiin vuonna 2000. Sen tarkoituksena oli sopia silloin vielä konseptivaiheessa olleelle .Net -ohjelmistokehykseen (engl. framework) ja tarkoituksena on olla helppo ja monikäyttöinen olio-ohjelmoinnin mahdollistava ohjelmointikieli. Sitä voidaan lisäksi käyttää esimerkiksi hajautetuissa järjestelmissä ja niiden ohjelmistokomponenteissa. Se ei kuitenkaan kilpaile tehokkuudessa tai kooditiedostojen koossa C tai Assembly -ohjelmointikielien kanssa. [30]

.Net on Microsoftin vuonna 2002 julkaisema yleiskäyttöinen kehitysalusta. Se mahdollistaa korkeantason ohjelmointiympäristön käytön ja se sisältää monipuoliset ominaisuudet. Lisäksi .NET tarjoaa mahdollisuuden muistinhallintaan ja rajapintoihin alemmalla tasolla .Net:ia voidaan käyttää yli 20 ohjelmointikielen kanssa Framework Class Library (FCL) -kerrosta hyödyntäen. Näistä suosituimpia ovat C#, F# sekä Visual Basic. .Net-alusta on laajentunut 2010-luvulla eri teknologia-alustoille ja nykyisin sitä voidaan käyttää mm. web-kehitykseen, mobiilisovelluksien kehitykseen ja pilvipalvelusovelluksiin. [31]

Työpöytäsovelluksen lisäksi yrityksessä on kehitetty sovelluksen web-versiota uuden 7.0 työpöytäversion ohella. Se sisältää täysin uudistuneen ulkoasun, sekä sisäisen toteutuksen. Web-selaimella käytettävä versio mahdollistaa responsiiviset erikokoisille ja resoluution näytöille skaalautuvat näkymät. Web-sovelluksen uusi versio ei vielä vuonna 2016 sisällä kaikkia työpöytäsovelluksen toimintoja. Web-sovelluksen ominaisuuksia on tarkoitus kehittää yrityksessä vastaamaan työpöytäsovelluksen toimintoja. Lisäksi yrityksen strategiana on siirtyä enemmän responsiivisiin web-sovelluksiin ja vähentää työpöytäsovelluksen tarvetta tulevaisuudessa. [23]

4.4 Web-käyttöliittymän nykyinen toteutus

SWD^{PES} -sovelluksen web-selainversio on kehitetty käyttäen Visual Studio ohjelmankehitysympäristöä. Sovellus koostuu palvelimilla suoritettavasta backend-puolesta ja käyttäjälle selaimessa toimivasta käyttöliittymästä. Palvelimella suoritettava backend-ohjelmakoodi käyttää ASP.NET:in MVC-arkkitehtuuria, sekä C#-ohjelmointikieltä. Selainkäyttöliittymä hyödyntää TypeScriptiä, joka toimii nykyisin lähes kaikissa uusimmissa selaimissa kuten Mozilla Firefoxissa ja Google Chromessa.

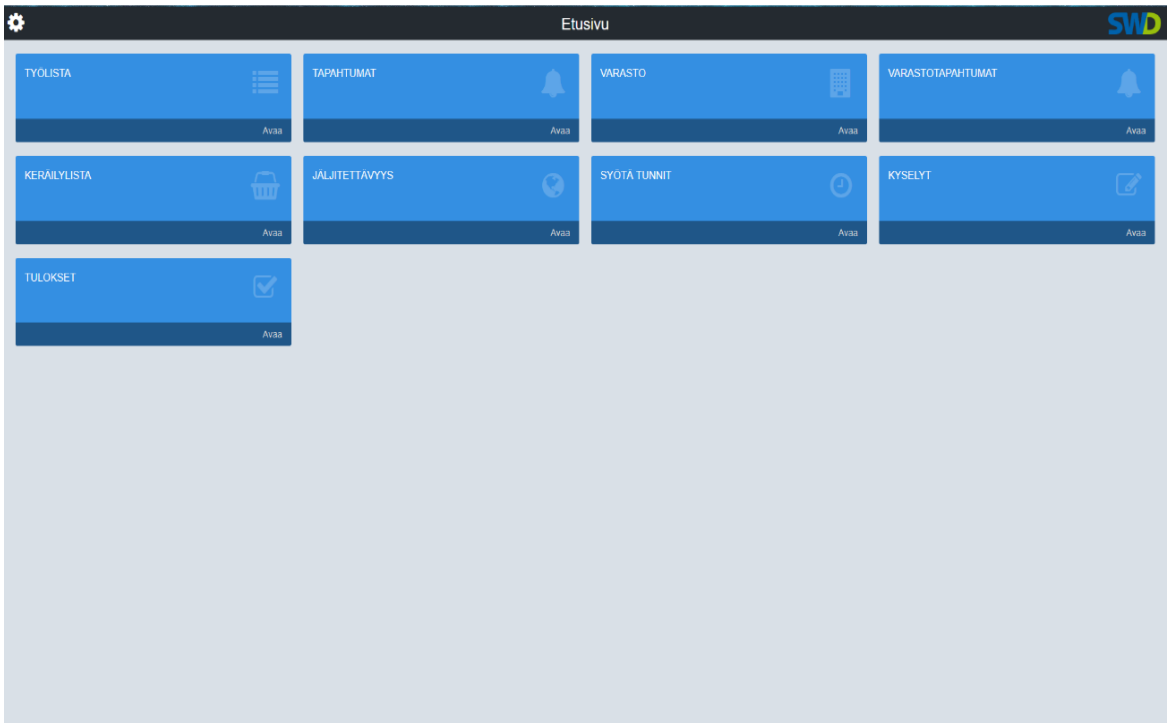
Sovelluksessa taustalla toimii Swd.pes.web -pääsovellus, joka yhdistää muut käytettävät komponentit yhdeksi kokonaisuudeksi. Sovellus toimii kuten SPA-sovellus ja hyödyntää

yrityksen omaa JavaScript-kirjastoa. Lisäksi käyttöliittymässä käytetään jQuery ja Bootstrap-kirjastoja. JQuery on monipuolinen JavaScript-kirjasto, joka mahdollistaa muun muassa helpon HTML ja DOM -manipuloinnin. JQueryn monipuolinen API-rajapinta sisältää myös metodit Ajaxin käyttöön, tapahtumien hallinnan (engl. event-handling) sekä animointien lisäämisen web-sivulle.

Bootstrap on JavaScript-kirjasto, jonka avulla on mahdollista tehdä moderneja web-käyttöliittymiä nopeasti. Sen hyötyihin kuuluu *mobile first*-idea, jossa web-sovelluksien suunnittelun lähtökohtana on tuottaa sovelluksia sopiviksi myös mobiililaitteiden ruuduille. Se tarjoaa monipuolisia kirjastoja käyttöliittämäelementtejä web-sovellukseen. Sekä Bootstrap että jQuery ovat avoimen lähdekoodin kirjastoja. [23]

Web-pääsovellus käyttää erilaisia itsenäisiä komponentteja, joista koostetaan sovelluksen varsinaiset toiminnallisuudet. Komponentit hyödyntävät yrityksen omaa *component base* -kirjastoa, joka sisältää web-kontrolleja. Näillä kontrolleilla pystytään toteuttamaan komponenttien ominaisuuksia. Yksittäiset komponentit ovat esimerkiksi tuntikirjaus-, varastot-, tai työjonokomponentti.

Komponentti on looginen kokonaisuus, joka yleensä hyödyntää ainoastaan itsensä ulkopuolelta vain *component base* -kirjaston tarjoamia valmiita kontrolleja. Se voi sisältää n-määrän erilaisia näkymiä, jotka voivat sisältää monia web-kontrolleja, kuten *list* tai *menu*-kontrolli. Ohjelmassa käyttöönotetut komponentit näkyvät päänäkymässä, jonka *Swd.pes.web* -pääsovellus sitoo kokonaisuudeksi. Pääsovellus osaa asetella komponentit responsiivisesti päänäkymään käyttöliittymässä (kuva 8). [23]



Kuva 8. *Esimerkki pääsivusta usealla komponentilla [23].*

Kuvassa 8 näkyy päävalikko SWD^{PES} -web-sovelluksessa. Sovelluksessa yksittäiset näkymät kuten työlista ovat kokonaisia moduuleja, joita voidaan lisätä käyttöön asiakkaan tarpeiden mukaisesti. Kuvan 8 päävalikon ulkoasua pystytään muokkaamaan esimerkiksi asiakkaan toiveiden mukaiseksi.

4.5 Tutkimuskysymys

Tämän diplomityön tutkimuksen tavoitteena on löytää SW-Development yritykselle sopiva JavaScript-kirjasto, jota voidaan käyttää responsiivisissä web-sovelluksissa.

Tässä työssä toteutetaan demosovellus, jonka avulla tutkitaan valittujen kirjastojen soveltuvuutta tuotantokäyttöön yrityksessä. Ennen tekniikoiden valintaa yritykseltä selvitettiin mahdollisia tarpeita sekä toiveita, joiden perusteella tekniikoita valittiin. Työssä on tavoitteena selvittää JavaScript-kirjastojen sopimista monipuolisen web-pohjaisen MES-järjestelmän toteuttamiseen. Lisäksi esiin tuli myös teknisiä rajoitteita valituille tekniikoille, kuten yhteensopivuus Microsoft Visual Studion kanssa.

Taulukossa 1 on koostetussa muodossa yrityksen tarpeita ja toiveita valituilta tekniikoilta. Taulukon 1 tietoja varten haastateltiin yrityksen teknisestä kehityksestä vastaavia henkilöitä. Lisäksi tietoja kerättiin yrityksen sisällä palaverikeskusteluissa.

Taulukko 1. *Yrityksen tarpeet valituilta tekniikoilta.*

Yrityksen tarve	Lisätiedot
Microsoft Visual Studio -yhteensopivuus	Yrityksen käytössä oleva kehitysympäristö. Microsoft Visual Studion pitää olla yhteensopiva tekniikan kanssa, muuten kehitys vaikeutuu. Käytännössä tekniikkakirjastojen pitäisi löytyä Microsoftin NuGet Gallery -paketinhallintajärjestelmästä.
Helppokäyttöisyys	Nopeasti opittavat tekniikat säästävät yrityksen resursseja, sekä mahdollistavat uusien työntekijöiden nopeamman kouluttamisen.
Hinta	Mahdollinen kirjaston hinta vaikuttaa, siihen otetaanko tekniikka käyttöön.
Suorituskyky	Tarkoitus sopia käytettäväksi mobiililaitteilla. Suorituskyky muutenkin tärkeää, koska esitettävät datamäärät voivat olla suuria. Esimerkiksi tuotannonsuunnittelunäkymän pitää pystyä esittämään jopa tuhansia osia samanaikaisesti.
Graafiset esitykset	PES-sovelluksessa on tarve esittää monia taulukoita, graafeja, gantt-näkymiä, sekä muita graafisia esityksiä. Uuden tekniikan olisi hyvä tarjota paljon valmiita ratkaisuja näiden kehitykseen. Tällä voidaan säästää huomattavasti aikaa, jos yrityksen ei tarvitse kehittää kaikkea itse.

Tekninen tuki ja hinta	Jos on tarjolla teknistä tukea, minkälaista tukea on saatavilla. Tuen hinta sovelluskehittäjälle.
Avoimen lähdekoodin tekniikoissa saatavilla oleva tuki	Avoimen lähdekoodin tekniikoissa suositaan yleisesti käytössä olevia tekniikoita, joihin löytyy mahdollisimman paljon valmiita ratkaisuja, sekä internetyhteisöjen keskustelua asiasta.
Vertailu nykyisiin tekniikoihin	Yrityksen toiveena on verrata valittuja tekniikoita nykyisin jo käytössä oleviin tekniikoihin. Esimerkiksi jQuery-kirjasto mainittiin tekniikkana, jota voitaisiin käyttää vertailupohjana.
Laajennettavuus ja yhteensopivuus muiden webtekniikoiden kanssa	Tarkoitus on myös selvittää valitun tekniikan tekninen arkkitehtuuri ja mahdollinen yhdistäminen muiden tekniikoiden kanssa. Lisäksi Onko kyseessä esimerkiksi kokonainen SPA-sovellusrunko vai pelkkä käyttöliittymäkirjasto.

5. JAVASCRIPT-OHJELMISTOKEHYKSET WEB-KÄYTTÖLIITTYMÄN KEHITYKSESSÄ

Moderneissa web-sovelluksissa käytetään JavaScriptiä eri toimintojen luomiseen. JavaScript ja sen päälle tehdyt valmiit JavaScript-kirjastot (engl. framework) ovatkin nousseet erittäin suosituiksi web-kehityksessä. Valmiiden kirjastojen avulla saadaan tuotettua moderneja web-sovelluksia paljon nopeammin kuin pelkän JavaScript ja HTML yhdistelmän avulla. Yleinen yhdistelmä on ollut käyttää Query-kirjastoa ja JavaScriptiä kompleksisien käyttöliittymien tekemiseen, mutta jQueryn lisäksi on olemassa todella monia valmiita kirjastoja.

Tässä luvussa selvitetään eri JavaScript-käyttöliittymäkirjastojen perusteita sekä vertaillaan niiden tarjoamia mahdollisuuksia ja käyttökohteita.

5.1 Tekniikoiden esittely

Seuraavissa alakohdissa esitellään erilaisia web-kehitykseen JavaScript-kielen kirjastoja, joita voidaan hyödyntää tässä työssä. Esitellyt kirjastot tarjoavat mahdollisuuden monipuolisten responsiivisten web-käyttöliittymien luomiseen.

Tätä diplomityötä varten selvitettiin monien eri JavaScript-kirjastojen ominaisuuksia, mutta linjattiin, että tutkitaan tarkemmin GitHub.com-versionhallintasivuston kolmea suosituinta käyttöliittymäkirjastoa (taulukko 2). Taulukossa 2 mainittu tähtien lukumäärä kertoo kuinka paljon yhteisön rekisteröityneet käyttäjät ovat antaneet suositus-merkintöjä kyseisille repositorioille. Kyseiset kirjastot ovat laajasti käytössä eri sivustoilla ja niille löytyy kattavaa dokumentaatiota internetistä. Ne myös tarjoavat monipuolisia graafisia komponentteja, joita voitaisiin hyödyntää yrityksen tuotteissa.

Taulukko 2. *GitHub.com-palvelun suosituimmat JavaScript-kirjastot [32].*

Nimi	Tähdet	Päivitetty viimeksi
twbs/bootstrap	106 188	Tammikuu 2017
facebook/react	58 669	Tammikuu 2017
angular/angular.js	54 505	Tammikuu 2017

Edellä mainittujen lisäksi selvitettiin yrityksessä nykyisin käytössä olevan ComponentOne ja jQuery -kirjastojen tarjoamia ominaisuuksia.

5.1.1 Bootstrap

Avoimen lähdekoodin Bootstrap-viitekehys on ollut pitkään suosittu JavaScript-kirjasto responsiivisilla web-sivustoilla. Bootstrap julkaistiin GitHub.com-sivustolla vuonna 2011, ja se oli vuonna 2014 sivuston suosituin projekti. Viitekehys tarjoaa monipuoliset ominaisuudet ja hyvän yhteensopivuuden eri web-selaimille sekä alustoille. Bootstrap sisältää HTML- ja CSS -templateja muun muassa fonttien, dokumenttien (engl. form), animaatioiden sekä navigaation luomiseen web-sivustolle. Bootstrap on tyypiltään modulaarinen kirjasto, joka koostuu SASS (Syntastically Awesome Stylesheets) -tyyppisistä (vanhemmissa versioissa LESS) dokumenteista, jotka kuvaavat erilaisia komponentteja. SASS on skriptikieli, joka kääntyy CSS-dokumenteiksi. Tämän lisäksi Bootstrap tukee monia muita JavaScript-lisäosia. [33]

Bootstrap on tarkoitettu helppokäyttöiseksi käyttöliittymäkirjastoksi, joka mahdollistaa web-kehityksen nopeasti. Se mahdollistaa helpon sekä tehokkaan tavan luoda skaalautuvia web-sivustoja ja sovelluksia, jotka sopivat eri näyttökoon laitteille. Skaalautuvuus toimii yhdellä koodipohjalla, jolloin kehittäjän ei tarvitse ylläpitää useita eri tiedostoja esimerkiksi mobiilikäyttäjille ja työpöytätietokoneen käyttäjille. [33]

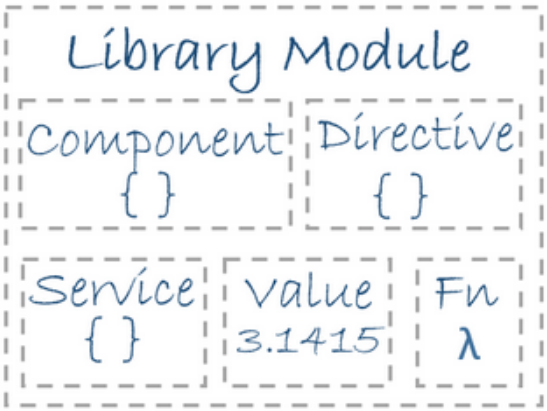
Bootstrap on mahdollista lisätä websivustolle suoraan valmiina tiedostona tai sen voi lisätä erilliseltä CDN (Content Delivery Network) -sivustolta. Jos Bootstrap lisätään CDN-sivuston avulla, voi käyttäjän selain hyödyntää mahdollisesti aikaisemmin ladattua tiedostoa välimuistissa. Tämä onnistuu, jos käyttäjä on avannut jo toisen sivuston, joka käyttää samaa Bootstrap-versiota. Tällöin sivun latausaika lyhenee. Käyttäjän ladatessa tiedoston CDN-sivustolta sivustot osaavat valita käyttäjää lähimmät palvelimet jolloin päästään lyhyempiin latausaikoihin. [33]

5.1.2 Angular 2

Angular (aiemmin AngularJS) on erittäin suosituksi noussut avoimen lähdekoodin JavaScript-kirjasto, jonka alun perin kehitti Google. Google julkaisi siitä ensimmäisen version vuonna 2012. Uudempi Angular 2 -versio julkaistiin syksyllä 2016. Uusi versio oli suuri muutos aikaisempiin versioihin verrattuna, eivätkä aikaisemmat versiot ole yhteensopivia version 2 kanssa. Julkaisun yhteydessä nimi AngularJS lisäksi lyhennettiin muotoon Angular. Puhuttaessa 1.x -versioista voidaankin käyttää AngularJS-nimitystä ja 2.x -versioista puhuttaessa nimeä Angular. [34]

Angularilla voidaan toteuttaa monimutkaisia SPA-sovelluksia, joiden suoritus tapahtuu käyttäjän selaimessa. Angular-sovellukset perustuvat model-view-controller (MVC) tai model-view-view model (MVVM) -arkkitehtuurimalleihin. Angular osaa automaattisesti käsitellä datan synkronisoinnin käyttöliittymäelementtien sekä JavaScript-objektien välillä (engl. 2-way data binding). Angularin 2 -versiossa siirryttiin käyttämään moduuleja, joita voidaan lisätä käyttöön tarvittaessa. Aikaisemmissa versioissa Angularin kaikki

ominaisuudet tulivat aina Angularin ytimen mukana. Siirtyminen moduuleihin pienensi kirjaston kokoa ja paransi sen suorituskykyä. Kuvassa 9 on esitelty Angular 2 -kirjastomoduuli ja sen sisältö. Kuvan mukaisia moduuleja voidaan asentaa Angulariin paketinhallintatyökaluilla, kuten Node package managerilla (npm). [34]



Kuva 9. *Angular 2 -kirjastomoduuli [35].*

Kuvassa 9 esitettävä Angular-sovelluksen moduuli koostuu *component-*, *directive-*, *service-*, *values-* ja *fn* -osista. *Component* on kontrolleriluokka, joka sisältää yleensä yhden käyttöliittymässä olevan näkymän (*view)* ja sen logiikan. Se voi olla esimerkiksi navigointivalikko käyttöliittymässä. *Directive* tarkoittaa määrittelyohjetta, joka voi sisältää esimerkiksi värityksen tiedot moduulin näkymille. *Service* tarkoittaa ulkopuolista palvelua, funktiota tai luokkaa, jota moduuli hyödyntää toiminnassaan. Yleensä *service* on tarkasti määritelty tekemään tiettyä asiaa, jonka se suorittaa tehokkaasti. Kyseisiä palveluita voi olla esimerkiksi lokitietoja keräävä palvelu, viestin välityspalvelu tai tietokantapalvelu. Moduulissa oleva *Fn* (kuva 9) tarkoittaa moduulin omia funktioita. Lisäksi moduuli sisältää kuvassa mainitut *value*-tiedot, jotka ovat moduulin sisältämiä muuttujia ja niiden saamia arvoja. [34]

Angularin suunnittelussa on panostettu modulaarisuuteen ja monipuolisiin testaustyökaluihin. Web-kehittäjälle tämä tarjoaakin monia muita web-tekniikoita paremmat mahdollisuudet testata sovellusta sekä etsiä kehityksenaikaisia virhetilanteita. Angularin oma *TestBed*-testaustyökalu sisältää testausmoduulin, jota voidaan konfiguroida testattavan asian mukaisesti. Sillä voidaan luoda testiympäristö, jossa testattava moduuli voidaan testata yksittäin, ilman muita sovelluksen moduuleja. Angularin omien testien lisäksi Angular-sovellusta voidaan testata muilla testityökaluilla kuten Jasminella, Karmalla tai Protractor -työkaluilla. Jasmine on JavaScript-koodin testaustyökalu. Karma mahdollistaa web-palvelimella tapahtuvan sovellustestauksen ja tapahtumalokin keräämisen. Protactor on *end-to-end* -testaustyökalu, jolla voidaan testata sovelluksen komponenttien yhteensopivuutta, sekä sovelluksen toimintaa alusta loppuun suoritettaessa. [34]

5.1.3 JQuery ja JQueryUI -kirjastot

JQuery ja sen eri versiot (mm. jQueryUI) ovat monipuolisia ilmaisia avoimen lähdekoodin JavaScript-kirjastoja. Sen ensimmäinen versio julkaistiin jo vuonna 2006 ja siitä on tullut suosituin JavaScript kirjasto web-sivustoilla [36]. Alkuperäisen version lisäksi jQuerysta on kehitetty myös muita versioita, kuten jQuery user interface (jQuery UI) sekä jQuery mobile. JQuery UI on tehty alkuperäisen kirjaston päälle ja se tarjoaa web-sivun käyttöliittymää varten monia efektejä, interaktio mahdollisuuksia, widget-tyyppisiä sovelluksia sekä teemoja. Mobile-versio jQuery:sta tarjoaa HTML5:een perustuvia käyttöliittymäominaisuuksia, joita voidaan käyttää mobiililaitteille tarkoitetuilla web-sivustoilla. [37]

JQuery on perustaltaan DOM-manipulointiin sopiva JavaScript-kirjasto. Käytännössä sen tarkoituksena on helpottaa HTML-modifiointia ja animointia web-sivustoilla. Se helpottaa myös AJAX-tekniikoiden käyttöä omalla rajapinnallaan, joka tukee useimpia nykyaikaisia selaimia. JQuery mahdollistaa myös monipuolisen tapahtumien käsittelyn, sekä erilaiset *event listener* -tyyppiset operaatiot, joita hyödynnetään käyttäjäinteraktion hallinnassa. Kuvassa 10 on jQueryn *click*-funktion esimerkki, jossa poimitaan käyttäjän tekemä klikkaaminen elementtiin ”p”. [37]

```
1
2   // Event setup using a convenience method
3   $( "p" ).click(function() {
4       console.log( "You clicked a paragraph!" );
5   });
```

Kuva 10. *JQueryn click()-funktio esimerkki.*

Kuvassa 10 aloitetaan jQueryn syntaksin mukaisella *$*-merkinnällä jQuery-koodi, jossa esitellään *click*-eventin jälkeen suoritettava lokin kirjoitus konsoliin. JQueryn versio 3 sisältää yli 60 *click*-funktion tyyppistä tapahtumien hallintaan liittyvää funktiota. Näitä ovat esimerkiksi *resize*, *mousedown* ja *keypress*-funktiot. [37]

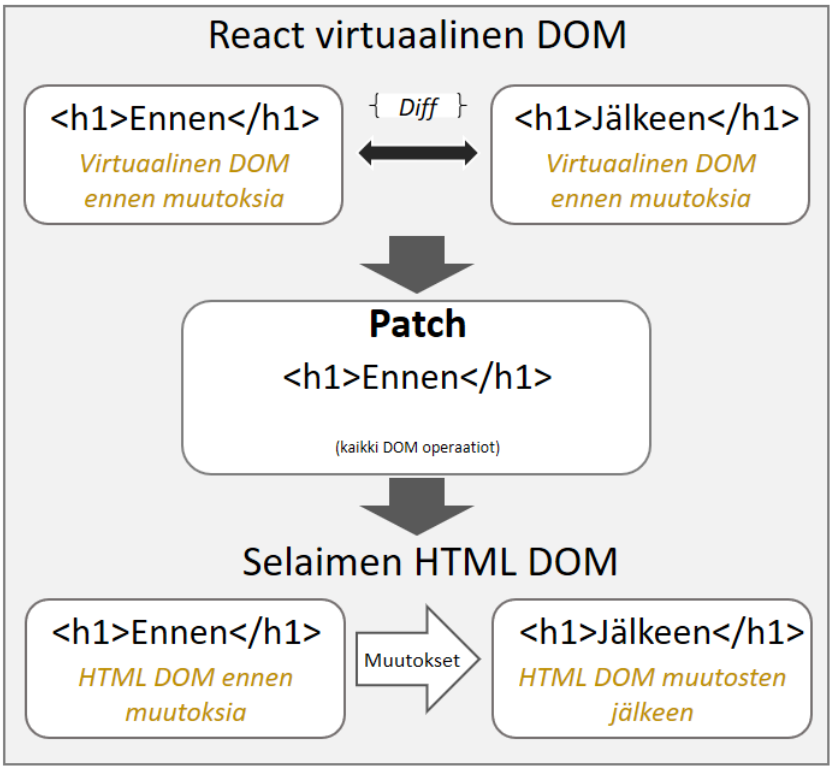
JQuery on julkaisunsa jälkeen ollut suosituimpia JavaScript-kirjastoa, koska sillä on saatu muun muassa parempi yhteensopivuus eri web-selainten kanssa. Varsinkin Internet Explorer -selaimen versio 8 ja sitä aikaisemmat versiot ovat usein tarvinneet omaa ohjelmakoodia, jotta web-sovellukset on saatu toimimaan kuten muilla web-selaimilla. Web-selaimet ovat kuitenkin kehittyneet jQueryn vuoden 2006 julkaisun jälkeen. Modernit selaimet, kuten Firefoxin versio 50, Chrome versio 55 ja Microsoftin Edge, tarjoavat kuitenkin samat tekniset ominaisuudet, jotka on ennen saatu web-sovellukseen jQuerylla.

Lisäksi modernit web-sovellukset toimivat nykyaikaisilla selaimilla ilman, että niitä tarvitsee muokata selainkohtaisesti. Tästä johtuen JQuery ei nykyisin ole yhtä tarpeellinen kuin silloin kun käytössä oli vanhempia web-selaimia. [38; 39]

5.1.4 React

React (myös React.js nimitystä käytetään) on avoimen lähdekoodin JavaScript-kirjasto, jonka on kehittänyt Facebook [40]. Kehitystyön jatkuessa mukana on ollut lisäksi avoimen lähdekoodin yhteisö. React on deklaratiivista paradigmaa hyödyntävä, tehokas, komponentteihin perustuvan JavaScript-kirjasto, jolla voidaan luoda interaktiivisia käyttöliittymiä web-sivustoille. React julkaistiin vuonna 2013, ja kirjoitushetkellä uusin vakaa versio on 15.3.2. Se on noussut nopeasti suosituksi eri web-sivustoilla ja esimerkiksi GitHub-sivustolla se listataan marraskuussa 2016 kuudenneksi eniten tähtiä keränneenä projektina [32].

React on tehokas SPA-sovellusten käyttämässä DOM-rakenteen (kohta 2.3) muokkaamisessa. Usein SPA-sovellukset muokkaavat HTML DOM:ia suoraan (esim. jQuery), mutta sen puutyyppisen rakenteen vuoksi tämä ei ole tehokasta verrattuna muuhun JavaScriptin suoritukseen. Reactissa perusideana on luoda virtuaalinen DOM-rakenne (engl. virtual DOM), jota voidaan muokata nopeammin kuin HTML DOMia. Virtuaalisesta DOMista pidetään muistissa kaksi versiota, joissa toinen on alkuperäinen ja toinen on päivitetty versio. Nämä kaksi annetaan Reactin funktioille, joka tarkistaa mahdolliset muutokset päivitetyssä versiossa. Tämän jälkeen kaikki muutokset päivitetään virtuaaliseen originaaliversio-DOMiin, jonka jälkeen päivitetty DOM siirretään myös HTML DOMiin. Kuvassa 11 esitetään Reactin DOMin päivittämisprosessi. Virtuaalinen DOM mahdollistaa Reactin hyvän suorituskyvyn verrattaessa muihin JavaScript-kirjastoihin. [40]



Kuva 11. *Reactin virtuaalinen DOM [41].*

Reactin kanssa käytetään yleensä myös Flux-arkkitehtuuria, joka on suunniteltu yhdessä Reactin kanssa. Flux on käytössä muun muassa Facebook.com -sivustolla. Flux mahdollistaa helposti ylläpidettävän dynaamisen käyttöliittymän ja datan hallinnan websovelluksessa. Käytännössä Flux mahdollistaa helpon yksisuuntaisen datan siirron, jossa hyödynnetään *event*-tapahtumakäsittelijöitä. [40]

Reactista on tarjolla lisäksi ReactJS.NET -versio, joka helpottaa Reactin käyttöä C#:n ja muiden .NET:iin liittyvien ohjelmointikielien kanssa. Se keskittyy erityisesti ASP.NET MVC web-kehitysympäristön yhteensopivuuteen. Uusimmat versiot tukevat ASP.NET:in versioita 4 ja 5. Lisäksi React tukee uudempaa ASP.NET Corea. ASP.NET Core on Microsoftin uusi avoimeen lähdekoodiin perustuva alustasta riippumaton viitekehys. [42]

5.1.5 ComponentOne-kirjastot

ComponentOne on GrapeCity-yrityksen tarjoama monipuolinen käyttöliittymäkirjasto. ComponentOne-kirjasto on SWD:llä jo aikaisemmin käytössä Windows Forms -työpöytäsovelluksissa ja se tarjoaa monipuoliset ominaisuudet myös web-kehitykseen. Lisäksi kirjasto on suunniteltu toimimaan yrityksellä käytössä olevan Visual Studio -kehitysympäristön kanssa. [43]

ComponentOne-kirjastoon kuuluu muun muassa MVC edition -tuote, joka sopii hyvin SPA-sovelluksiin. MVC edition tarjoaa mobiilikäyttöön ja kosketusnäytöille sopivia taulukoita sekä ruudukkotyökaluja. ComponentOne tarjoaa myös ASP.NET Web forms, HTML5, Silverlight sekä Lightswitch -teknologioihin perustuvia kirjastoja. [43]

ComponentOne-kirjasto on maksullinen,eikä se ole avoimen lähdekoodin kirjasto. ComponentOne kuitenkin tarjoaa laajan dokumentaation tuotteilleen ja tarvittaessa tukipalveluita yrityksille. SWD:llä on myös kokemusta kirjaston käytöstä nykyisissä .Net-sovelluksissa ja websovelluksissa.

5.2 Tekniikoiden vertailu

Tässä työssä tutkittuja tekniikoita on verrattu paljon myös kirjallisuudessa. Hannahin artikkelissa selvitettiin neljän eri JavaScript-viitekehyksen paremmuutta. Artikkelin vertailussa oli mukana Backbone, AngularJS, Ember ja React -viitekehykset. Hannah kertoo Angularin sopivan moniin projekteihin, mutta sen suorituskyky ei ole Reactin tasolla. [44]

Angularin hyviksi puoliksi artikkelissa lasketaan kaksisuuntainen datan muokkaus (engl. two-way data binging), direktiivit, sekä helppo *service*-palvelujen lisääminen Angular-moduuleihin. Myös Googlen tuki todetaan olevan hyväksi viitekehykselle. Huonoja puolia Angularissa todetaan olevan suorituskyky, palvelimella suoritettavan HTML-dokumentin luominen, sekä suuri opettelun määrä kehityksen aloituksessa. Lisäksi kaksisuuntainen datan muokkaus mainitaan myös huonona puolena, koska se vaikeuttaa debuggausta ja huonontaa viitekehyksen suorituskykyä. [44]

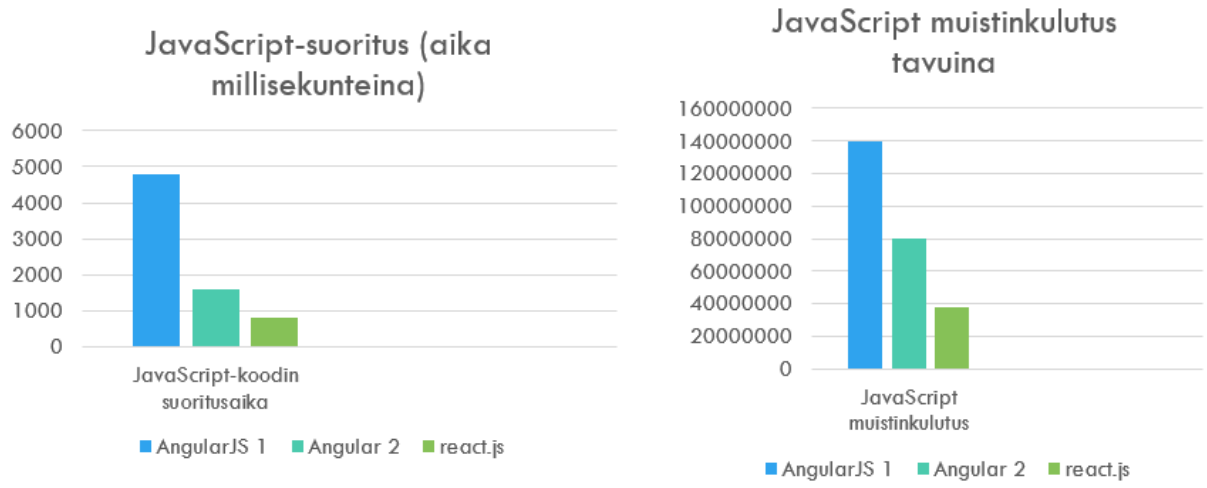
Hannahin artikkelissa kerrotaan, että Reactia käytetään muun muassa facebook.com-, instagram.com ja netflix.com -sivustoilla. Artikkelin mukaan se sopii hyvin kaiken kokoisiin projekteihin ja suorittaa SPA-sovellusten tärkeimmät ominaisuudet hyvin. React on Hannahin mukaan suorituskyvyiltään kaikkein paras vertailun kirjastoista. Sen mahdollistaa virtuaalinen DOM, jota React hyödyntää. React tiedosto pakattuna (120 Kt) on kuitenkin kaikkein suurin vertailun kirjastoista, mutta sillä ei ole riippuvuuksia muihin tiedostoihin. [44]

Reactin etu on Hannahin mukaan sen opettelun helppous. Angular- ja Ember-kirjastoihin verrattuna React on helppokäyttöinen, koska se käyttää pelkästään Reactin käytössä olevaa JSX-ohjelmointikieltä. Jos kehittäjä osaa valmiiksi JavaScriptiä, on Hannahin mukaan Reactin ja JSX:n käyttö opittavissa nopeasti. Lisäksi Reactin komponentteihin perustuva ohjelmointiparadigma on lähellä muita JavaScriptiin pohjautuvia ratkaisuja, kuten CommonJS-moduuleja. Komponentteja voidaan myös käyttää uudelleen, joka mahdollistaa nopean ohjelmoinnin. Reactilla on myös mahdollista käyttää palvelimella tapahtuvaa HTML-dokumentin luomista, jota esimerkiksi Angular ei tue. Tämä poistaa useita

suorituskykyyn liittyviä ongelmia, joita voi esiintyä, kun käyttöliittymän HTML-dokumentti luodaan ja modifioidaan käyttäjän selaimessa. [44]

Hannah myös mainitsee Reactin etuna mahdollisuuden luoda mobiilisovelluksia React Nativen avulla. Tällöin kehittäjän tarvitsee opetella vain kerran Reactin käyttö ja sen jälkeen voi ohjelmoida sekä web-sovelluksia että iOS ja Android -mobiiliapplikaatioita. Käytännössä React native käyttää JavaScriptiä ja Reactia, joiden avulla kehitetään sovelluksia suoraan Android ja iOS -alustoille. Toisin sanoen yksi ohjelmakoodi ei suoraan käänny kaikille alustoille, mutta Reactia käyttämällä voi luoda usealle alustalle natiivisovelluksia. Hannahin artikkelissa vertailun parhaaksi kirjastoksi valitaan React. Sen suorituskyky, helppokäyttöisyys ja monipuolisuus nostavat sen muiden vertailun kirjastojen edelle. [44]

JavaScript-kirjastojen suorituskykyä on vertailtu Peyrottin artikkelissa. Artikkelissa JavaScript-kirjastoilla suoritettiin useita erilaisia suorituskykytestejä. Kuvassa 12 esitellään AngularJS 1, Angular 2, ja React -kirjastoille Peyrottin artikkelissa tehtyjen suorituskykymittausten tulokset. Mittaustuloksista nähdään, että JavaScript-koodin suoritusaikaa mitattaessa React suoriutuu yli 50% nopeammin kuin Angular 2. Lisäksi vanhempi AngularJS 1 -version suoritusaika on lähes 5000 millisekuntia, kun React jää alle 1000 millisekuntiin. [45]



Kuva 12. *Suorituskyky mittauksia JavaScript-kirjastoilla [42].*

Artikkelissa mitattiin myös kirjastojen muistinkulutusta JavaScript-ohjelmakoodin suorituksen aikana. Angularin versio 2 käyttää alle 60% muistia selaimessa verrattuna vanhempaan AngularJS 1 -versioon. React kuluttaa myös Angular 2:een verrattuna yli 50% vähemmän muistia. Muistinkulutuksen merkitys kasvaa, kun SPA-sovellusta käytetään mobiililaitteilla, koska mobiililaitteiden suorituskyky ja muistin määrä ovat yleensä pienempiä kuin työpöytätietokoneilla. Peyrottin artikkelissa Reactin kerrotaan sisältävän hyvän suorituskyvyn sekä helpon integraatiopolun. Lisäksi React tuottaa yksinkertaista ja helposti ymmärrettävää koodia. [45]

Reactilla on myös laaja kehittäjäyhteisö, josta löytyy nopeita ratkaisuja moniin ongelmiin. Angular 2:n mainitaan olevan hyvä laajoissa sovelluksissa, mutta Angularin arkkitehtuuri on kuitenkin työläs opetella. Angular 2 on tehty käytettäväksi TypeScript-ojelmointikielen kanssa ja se voidaan laskea eduksi, jos TypeScript-kieli on kehittäjälle tuttu muista sovelluksista. [45]

5.3 Yhteenveto tekniikoista

Tässä työssä vertailtiin JavaScript-kirjastoja. Vertailuun valittiin Bootstrap, Angular 2, jQuery, React, ComponentOne -kirjastot. Taulukossa 3 on listattu kaikki edellä mainitut tekniikat ja listattu niiden eroavaisuuksia, ominaisuuksia ja ominaispiirteitä.

***Taulukko 3**. Käyttöliittymäkirjastojen vertailu.*

Nimi	Lähdekoodi	Visual Studio yhteensopivuus	Tekninen tuki yrityksille	Arkkitehtuuri	Mobiililaitteille optimoitu	Muuta
Bootstrap	Avoin	Toimii hyvin	Ei saatavilla	Front-end -viitekehys	Kyllä	Sopii responsiiville sivustoille.
Angular 2	Avoin	Visual Studio 2015 Update 3 ja uudemmat	Ei saatavilla.	Tarjoaa valmiin rungon SPA-sovelluksiin	Kyllä	Sopii responsiiville sivustoille.
jQuery	Avoin	Toimii hyvin	Ei saatavilla	Yleiskäyttöinen JavaScript-viitekehys DOM-manipulaatioon		Yrityksellä käytössä

React	Avoin	Toimii hyvin	Ei saatavilla	View-osa MVC-mallissa	Toimii mobiililaitteilla	Suorituskyky valituista tekniikoista paras. Sopii responsiiville sivustoille.
ComponentOne	Ei avoin	Toimii	Saatavilla	MVC-malli	Kyllä	Yrityksellä käytössä

Taulukossa 3 listattujen ominaisuuksien lisäksi tutkittiin, löytyykö kyseisille JavaScript-kirjastoille valmiita graafisia elementtejä, joita voidaan hyödyntää yrityksen MES-tuotteissa.

Angularille on saatavilla ilmaisena monia taulukkotyökaluja. Yrityksen tuotteessa käytössä oleva gantt-kaavio, jota hyödynnetään aikatauluttamisessa, oli saatavilla ainakin Angular-gantt-liitännäisenä MIT-lisenssillä angular-gantt.com-sivustolla. Lisäksi monia yrityksen nykyisessä PES-tuotteessa käytettäviä kaavioita on saatavilla vapaana lähdekoodina github.com-sivustolla, kuten Angular Chart -kirjaston kaavioina. [46]

Myös React-kirjastolle on tarjolla useita gantt-kaaviototeutuksia. NPM-paketinhallintasivusto (www.npmjs.com) tarjoaa ilmaisen React-gantt-paketin, joka mahdollistaa gantt-kaavioiden esittämisen. Myös Gantt-for-react-kirjasto on saatavilla NPM-sivustolta ilmaisena. Kaavioiden ja taulukoiden esittämiseen Reactilla löytyy React 3D -kirjasto, joka tarjoaa satoja valmiita kaaviopohjia. Kirjasto tarjoaa myös laajan dokumentaation kirjastosta löytyvien kaavioiden käyttämiseen sovelluksissa. [47]

Bootstrap-kirjasto tarjoaa valmiita ulkoasukomponentteja, kuten painike-ikoneita, valikkopalkkeja, pudotusvalikkoja, *progress bar* -elementtejä, sekä paneelielementtejä. Bootstrap ei kuitenkaan itsessään sisällä valmiita toteutuksia esimerkiksi gantt-kaavioille tai taulukoille. Bootstrap voidaan kuitenkin yhdistää muiden tekniikoiden kanssa. Esimerkiksi Reactia varten on olemassa React-Bootstrap -kirjasto, jossa Bootstrap on kirjoitettu kokonaan uudelleen Reactia varten [40]. Se mahdollista uudelleenkäytettävät Bootstrap-käyttöliittymäkomponentit, joita voidaan käyttää Reactin kanssa.

5.4 Demon toteutukseen valitut tekniikat

Tässä työssä selvitettiin viiden eri JavaScript-kirjaston soveltuvuutta SW-Development-mentille. SW-Development käyttää vuonna 2017 muun muassa jQuery ja ComponentOne -kirjastoja. Nämä otettiin vertailuun mukaan, mutta niitä ei valittu työssä toteutetun sovellusdemon toteutukseen. Yrityksellä oli paljon aikaisempaa kokemusta ja informaatiota näistä tekniikoista. Tahtotilana olikin tutkia, saataisiinko mahdollisesti muilla tekniikoilla parempi tuloksia, kun yhtiön tuotteita kehitetään jatkossa enemmän web-sovelluksina.

Bootstrap-kirjaston käyttöliittymäelementtejä voidaan myöhemmin tarvittaessa lisätä web-sovellukseen. Tässä työssä tehdyn demosovelluksen kehitysvaiheessa Bootstrap-kirjastoa ei lisätty sovellukseen. SW-Developmentilla oli jo aikaisempaa kokemusta Bootstrap-kirjastosta ja sen ominaisuuksista, sillä Bootstrap on käytössä tietyissä yrityksen projekteissa.

Angular 2 ei tullut valituksi demosovelluksen tekniikaksi useasta syystä. Angularin uudempi 2-versio julkaistiin virallisesti vasta tutkimuksen aikana, eikä siitä ollut saatavilla riittävästi tietoa, jotta Angularia voitaisiin käyttää yritystuotteissa. Angular 2:n suorituskyky on parantunut aikaisempiin versioihin verrattuna, mutta kuten kuvassa 12 nähdään, Angular 2 häviää selvästi suorituskykyvertailussa React-kirjastolle. Angular 2 myös tarjoaa paljon työkaluja kehityskäyttöön, mutta Hannahin artikkelissa ja Codementor-sivuston artikkeleissa mainitaan, että Angular 2 käyttäminen on aloittelijalla työläämpää kuin React-kirjaston käyttäminen. Lisäksi kirjastolle ei ollut vielä saatavilla laajaa kehittäjien tukea internetsivustoilla. Kauemmin valmiina olleelle React-kirjastolle oli esimerkiksi Stackoverflow.com -sivustolla enemmän kysymyksiä ja vastauksia. Angular 2 -versiolle oli kuitenkin tarjolla paljon erilaisia taulukko ja kaaviopohjia, joita oltaisiin voitu hyödyntää yrityksen tuotteissa. [44; 48]

Työssä valittiin käytettäväksi React-kirjastoa käyttöliittymän kehittämisessä. Reactin suorituskyky oli vertailluista kirjastoista paras. Kuten kuvasta 12 nähdään, oli React yli 50% nopeampi JavaScript-koodin suorituksessa kuin Angular 2. Molemmat edellä mainituista kirjastoista olivat myös tehokkaampia kuin vanhempi AngularJS 1 -kirjasto. Suorituskyky on tärkeää yritykselle, sillä sen tuotteissa esitetään suuria datamääriä yksittäisissä elementeissä. Esimerkiksi gantt-näkymässä esitetään usein jopa tuhansia yksittäisiä liikuteltavia gantt-palikoita yhtäaikaisesti.

Yrityksellä on myös tavoitteena saada tulevaisuuden sovelluksistaan responsiivisia ja mahdollisimman käyttäjäystävällisiä. Tämä tarkoittaa, että responsiivisen web-sovelluksen pitää toimia sulavasti myös esimerkiksi tablet-tyyppisillä laitteilla ilman, että sovellus jää lataamaan näkymiä pitkiksi ajoiksi. React mainittiin myös monessa lähteessä helppokäyttöisemmäksi kuin esimerkiksi Angular 2 -kirjasto [44; 48]. Tämä oli yritykselle tär-

keä kriteeri, koska työntekijöiden koulutus alustaan on tällöin nopeampaa. Helppokäyttöisyys ja hyvä koodin ymmärrettävyys auttavat myös tuotteiden ylläpidossa ja virheiden korjauksessa.

6. DEMOSOVELLUKSEN TOTEUTUS VALITUILLA TEKNIIKOILLA

Tässä luvussa käydään läpi web-sovelluksen toteuttamisen vaiheet. Työssä toteutetaan kvalitatiivinen tutkimus, jonka tarkoituksena löytää yritykselle sopiva käyttöliittymäkirjasto.

Luvussa käydään läpi ensiksi tutkimuksen tavoitteet ja arviointikriteerit, joilla toteutettua sovellusta arvioidaan. Lisäksi käydään läpi sovelluksessa käytettävät tekniset ominaisuudet. Tutkimuksen sovelluksen tarkoituksena on tutkia valittujen teknologioiden sopimista yrityksen tarpeisiin. Sovellusta ei sellaisenaan käytetä yrityksen SWD^{PES}-tuotteessa, mutta toteutusta voidaan tarvittaessa hyödyntää, kun yrityksen tuoteportfoliota kehitetään tulevaisuudessa.

6.1 Tavoitteet ja arviointikriteerit

Tutkimuksessa toteutetaan responsiivinen demosovellus, joka koostuu dynaamisesti muokattavasta puurakenteesta. Tavoitteena on selvittää, miten valittu React-käyttöliittymäkirjasto sopii demosovellukseen, ja voitaisiinko Reactia käyttää laajemminkin yrityksen tuotteissa. Yritys on tuotekehityksessä selvittänyt Reactin käyttöä, ja toiveena oli saada siitä tarkempaa informaatiota.

Yritys on tuottamassa tulevaisuudessa useita ratkaisuja web-teknologioihin perustuen ja tavoitteena on siirtyä pääasiallisesti web-teknologioihin. Nykyisin monet yrityksen ratkaisuista käyttävät .NET ja Windows Forms -alustoja, joilla tuotetut sovellukset toimivat ainoastaan Windows-käyttöjärjestelmällä varustetuilla tietokoneilla. Yrityksen tuoteportfolioon on kuulunut myös tiettyjä web-sovelluksia, mutta pääsääntöisesti on käytetty työpöytäsovelluksia.

Arvioinnin pohjaksi otettiin yhteensopivuus yhtiön käyttämän Visual Studio -kehitysympäristön kanssa. Yrityksen web-sovellukset käyttävät Microsoftin MVC (model-view-controller) -mallia, joka tulee ASP.NET -kehyksen tarjoamana. Tässä työssä keskityttiin tekniikoihin, joita voidaan hyödyntää MVC-mallin *view*-osuudessa. *View* eli näkymä tarkoittaa käyttäjän selaimessa pyörivää osuutta MVC-mallista.

Tärkeä kriteeri tekniikalle oli helppokäyttöisyys ja mahdollinen tekninen tuki. Helppokäyttöisyys säästää kehityskuluissa ja myös tuotteiden elinkaaren myöhemmissä vaiheissa tehtävässä ylläpidossa sekä jatkokehityksessä. Jos ohjelmakoodi on helposti ymmärrettävää ja sitä pystytään helposti myös muokkaamaan jälkikäteen, se auttaa yrityksen kannattavuudessa. Edellä mainittujen kriteerien lisäksi haluttiin tarkastella seuraavia asioita:

- Suorituskyky suurilla data määrillä.
- Mahdollisuus käyttää nykyisiä tietokantaratkaisuja, eli käytännössä yhdistää SQL-tietokantaan, joka hallinnoidaan Microsoftin SQL Server Management -työkalulla.
- Toiminta moderneilla selaimilla, esimerkiksi Google Chrome versio 50 ja sitä uudemmat versiota, Mozilla Firefox versio 50 ja uudemmat versiot, Microsoft Edge kaikki versiot, sekä Internet Explorer versio 11 ja uudemmat versiot.
- Soveltuminen responsiiviin sovelluksiin.

6.2 Arkkitehtuuri toteutukselle ja valitut tekniikat

Sovelluksena toteutettiin puutyyppinen tietorakenne, jonka tasot kuvattiin graafissa, jota käyttäjä pystyy muokkaamaan. Kehitystyöasemana käytettiin Microsoft Windows 10 käyttöjärjestelmää, jota suoritettiin kannettavalla tietokoneella. Kannettavan tietokoneen tekniset tiedot on listattu taulukossa 4.

Taulukko 4. *Kehityksessä käytetyn tietokoneen spesifikaatio.*

Prosessori	Intel Core i5-3337U (kellotaajuus 1.80 Ghz)
RAM	4.00 Gt
Grafiikkasuoritin	Intel HD Graphics 4000
Käyttöjärjestelmä	Microsoft Windows 10 Pro, Versio 10.0.14393

Sovelluksen koodauksessa käytettiin Visual Studio 2015 Enterprise -ohjelmankehitysympäristöä. Visual Studio käyttää web-sovellusten suorittamiseen IIS Express (Internet Information services) -sovellusta. IIS Express versiona oli 7.0. Lisäksi käytössä oli NPM-paketin hallinta sovellus, joka on integroitu Visual Studio 2015 -sovellukseen. Tietokantana sovelluksessa voi käyttää SQL-tietokantaa.

Kohdassa 5.4 valittiin MVC-mallin mukaisen näkymän toteutustekniikaksi React-JavaScript-kirjasto. Reactin kanssa käytettiin XML:ää muistuttavaa JSX-ohjelmointikieltä, joka kääntyy JavaScriptiksi. Taulukkoon 5 on koottu web-sovelluksessa käytettäviä tekniikoita.

Taulukko 5. *Web-sovelluksen tekniikat.*

Nimi	Versio	Muuta
React	15.4.0	
JSX	-	Reactin käyttämä ohjelmointikieli
Node	6.9.2	
CSS	CSS3	Tyylidokumentaatio
Visual Studio	2015 Enterprise	Kehitysympäristö
NPM	3.10.9	Paketinhallinta
IIS Express	7.0	Paikallinen web-palvelin
ASP.NET	5.2.3	MVC-sovelluspohja

Sovelluksen käyttöliittymän luomisessa käytettiin React-kirjastoa. Reactin kanssa käytettiin JSX-ohjelmointikieltä, joka on luotu Reactia varten. JSX korvaa tavallisen JavaScriptin web-sovelluksessa, mutta JSX käännetään lopuksi JavaScriptiksi, jota web-selaimet ymmärtävät. Sen ideana on olla esiprosessointivaihe, joka lisää XML:ää muistuttavan syntaksin JavaScriptiin. Käännösvaiheessa JSX-koodia optimoidaan ja sen tuottama JavaScript-koodi onkin tehokkaampaa kuin suoraan JavaScript-kielelle ohjelmoitu sovellus. JSX:n suurimmat edut ovatkin tehokkuus, turvallisempi muuttujien tyypitys, helpompi debuggaus web-kehityksessä, sekä helpompi *template*-mallien koodaaminen.

JSX:n syntaksi on yleensä lähellä XML:n merkkaustapaa, mutta siinä on kuitenkin useista eroja XML:ään verrattuna. Siinä käytetään Javaa muistuttavia luokka- ja funktiomäärittelyitä. Ohjelmassa 4 on esitelty JSX-luokka, jossa esitellään otsikko ”*Header*” sovellukseen. Kuvassa nähdään myös JSX:n kaarisulkeet ominaisuus, jolla voidaan kirjoittaa tavallista JavaScriptiä dokumentissa. Ohjelmassa 4**Virhe. Viitteen lähdettä ei löytynyt.** *var*-muuttuja *myStyle* on esitelty riveillä 4–6 ja sitä käytetään kaarisulkeiden sisällä rivillä 10.

JSX-ohjelmointikielessä v*ar*-muuttuja on samanlainen muuttuja kuin JavaScriptin *var*-muuttuja, jotka ovat identifioitu nimellä ja niihin voidaan tallentaa erilaisia datatyyppejä. Tässä tapauksessa se määrittelee tyylin ”*Header*”-tekstille, joka on *h1*-tason otsikkorivi (ohjelma 4).

```
1    class App extends React.Component {
2       render() {
3
4          var myStyle = {
5             fontSize: 100,
6             color: '#FF0000'
7          }
8          return (
9             <div>
10               <h1 style = {myStyle}>Header</h1>
11            </div>
12         );
13      }
14   }
15
```

Ohjelma 4. *JSX:llä kirjoitettu React-luokka.*

Ohjelmassa 4 on myös esimerkki *React.Component* -luokan käytöstä. *React.Component* on abstrakti luokka, josta voidaan periyttää aliluokkia. Kyseisiä aliluokkia voidaan käyttää itsenäisinä ja uudelleenkäytettävinä käyttöliittymän osina. Esimerkissä luokka *App* on periytetty *React.Component*-luokasta. Se sisältää *h1*-otsikkotekstin ja muotoilun tälle otsikkotekstille. Muotoilu on määritelty *myStyle*-muuttujassa. Muotoilu asetataan tekstille antamalla se *style*-kuvauksena.

Reactissa luokkia voidaan periyttää *extend*-määrittelyllä. Tällöin voidaan esimerkiksi esittää ”*Class Content extends React.Component*”. Tällä syntaksilla laajennettaisiin kyseistä *React.Component* -luokkaa. Kun *React.Component* -luokkaa kutsutaan muualta ohjelmakoodista, periytyy myös lisätty *Content*-luokka sen mukana.

6.3 Toteutus

Tutkimuksessa toteutettiin responsiivinen web-sovellus, jossa on puutyyppinen muokattava tietorakenne. Web-sovelluksen toteutus voitaisiin yhdistää yrityksen nykyisiin sovelluksiin lisäosana. Puurakenteella tarkoitetaan kaaviota, jossa eritasot on yhdistetty toisiinsa lapsi-vanhempi (engl. child-parent) linkillä. Näitä linkkejä voi olla ääretön määrä sekä allekkain, että sisennettyinä. Toisin sanoen tasoilla voi olla alatasoja riippumatta niiden omasta tasosta rakenteessa ja myös niillä voi edelleen olla alempia tasoja.

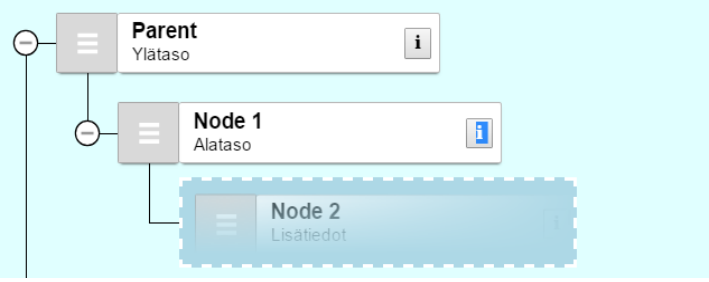
Puurakenteen ideana oli esittää yrityksen työpöytäsovelluksessa käytettävä puurakenne dynaamisena web-sovelluksena. Web-sovelluksen piti pystyä responsiivisuuteen, jolloin se toimii erikokoisilla näyttöruuduilla. Puurakennetta voidaan hyödyntää suurien rakenteiden kuvaamisessa. Tällainen rakenne voi olla esimerkiksi tuotannonohjausjärjestelmän tuoterakennenäkymissä.

Toteutetun web-sovelluksen puurakennetta pystyy muokkaamaan ja konfiguroimaan. Puurakenteen käyttöliittymässä havainnollistetaan puun eritasoja, alkioita ja niiden välisiä suhteita. Puun alkio koostuu otsikkotekstistä, lisätietotekstistä ja alkiolle asetettavista parametreista. Tasojen ja alkioiden väliset suhteet kuvataan viivoilla. Sovelluksessa olevia soluja voidaan lisätä, poistaa ja muokata niiden sijaintia puurakenteessa. Solujen sijainnin vaihtaminen tapahtuu valitsemalla alkio hiirellä ja raahaamalla alkio toiseen kohtaan puurakenteessa.

Puun yksittäisille soluille voidaan asettaa parametreja, jotka muokkaavat niiden ominaisuuksia. Esimerkiksi solulle voidaan asettaa kielto, joka estää lapsisolujen lisäämisen sen alle. Toinen parametri estää solun vanhemman vaihtamisen.

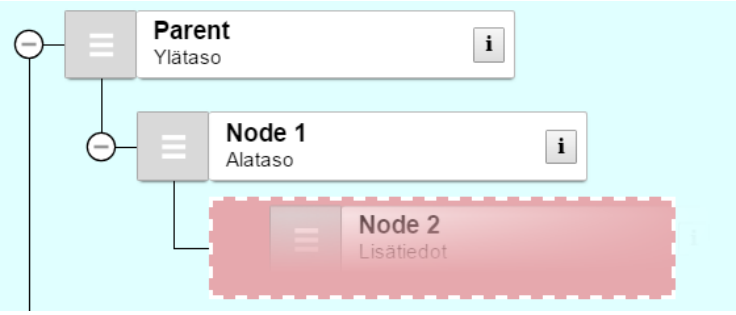
Puurakenteen tasoja voidaan laajentaa ja supistaa rivillä olevasta painikkeesta. Tämä painike muuttaa ikonia tilanteesta riippuen esittäen plus- ja miinusmerkkejä. Jos puun solussa ei ole alempia tasoja, se ei sisällä laajennus- tai supistusikonia. Puun tasoja käyttäjä pystyy siirtämään hiirellä liikuttamalla. Kun solua liikutetaan toiseen kohtaan, näkymä indikoi väreillä, onko solu mahdollista siirtää kyseiseen kohtaan. Jos solu pystytään siirtämään kyseiseen kohtaan, se näkyy sinisellä värityksellä. Jos solua ei saa siirtää kyseiselle tasolle, sen indikaationa näkyy punainen väri kyseisessä kohdassa.

Kuvassa 13 esitetään solun nimeltä Node 2 siirto Node 1 -nimisen solun alle, joka on sallittu siirto.



Kuva 13. *Sallittu siirto puurakenteessa.*

Kuvassa 14 esitetään tilanne, jossa Node 1 -nimiselle solulle on laitettu parametri, joka estää uudet alatasot. Lisäksi solulle voidaan asettaa parametriksi solun siirtämisen esto, jolloin solua ei voida siirtää eri tasolle tai toisen solun alle.



Kuva 14. *Kielletty siirto puurakenteessa.*

Työn sovelluksessa toteutettiin myös hakutoiminto, jolla pystyy etsimään puurakenteesta soluja niiden otsikkokentän perusteella. Hakutoiminnossa käyttäjä syöttää etsittävän tekstin, jonka perusteella etsitään puurakenteen soluista vastaavaa tekstiä. Tämän jälkeen hakutoiminto osaa näyttää löydettyjen solujen kokonaismäärän ja värittää solut eri värillä kuin muut solut. Lisäksi hakutoiminnolla voidaan kohdistaa näkymä löydettyihin soluihin ja siirtyä niiden välillä, jolloin käyttäjän ei tarvitse puurakennetta vierittämällä etsiä soluja.

6.4 Sovelluksen arviointi valituilla kriteereillä

Työssä toteutettu sovellus arvioidaan kohdassa 6.1 esitettyjen kriteerien mukaisesti. Ensimmäisenä kriteerinä oli pystyä käyttämään Visual Studio ohjelmointiympäristöä, joka yrityksellä on käytössä. Sovellus koodattiin Visual Studiota käyttäen ja tarvittavat tekniikat (React) voidaan ladata käyttöön suoraan Visual Studion omasta paketinhallinta järjestelmästä. Lisäksi React-sovellusta voidaan käyttää Visual Studion sisältämällä IIS Express web-palvelimella, joten sovellus on hyvin yhteensopiva yrityksen nykyisiin järjestelmiin.

Yrityksen tavoitteena on käyttää tekniikoita, joita voidaan kouluttaa helposti työntekijöille. Tekniikan helppokäyttöisyyttä arvioitiin subjektiivisesti, sekä verrattiin sitä kirjallisuuteen. Tutkimuksen tekijällä ei ollut aikaisempaa kokemusta React-tekniikasta ja kokemus web-kehityksestä oli vähäistä. Sovelluksen tekeminen oli kuitenkin nopeaa ja Reactin kanssa käytetty JSX-ohjelmointikieli nopeutti kehitystä, koska se sisältää muun muassa muuttujien tyypityksen. Lisäksi Reactille on saatavissa laaja dokumentaatio, sekä paljon kehittäjäyhteisön tukea esimerkiksi stackoverflow.com -internetsivustolla.

Sovelluksen kehityksessä käytettiin Google Chrome (versio 56.0.2924.76) -internetselainta. Lisäksi sovellus testattiin Mozilla Firefox (versio 52.0.4) ja Microsoftin Edge (versio. 38.14393.0.0) -selaimilla. Sovellus toimi kaikilla kyseisillä selaimilla.

Suorituskyvyn arviointiin käytettiin Chrome-selaimen tarjoamia kehittäjätyökaluja. Sovelluksen puurakenteeseen lisättiin solujen määrää ja mitattiin sivun latausaika. Lataus-

ajoista katsottiin sivun kokonaislatausaika (Chromessa: Total), sekä JavaScriptin suorittamiseen käytetty aika (Chromessa: Scripting). Latausajat mitattiin käyttämällä solujen määrinä 10, 100 ja 100 000 ylätason solua. Ylätason solut sisälsivät kaksi lapsi-solua. Mittaukset suoritettiin kaikilla määrillä viisi kertaa, ja mittauksista laskettiin keskiarvo suoritusajoille. Taulukossa 6 on esitelty tulokset suorituskyky mittauksille.

Taulukko 6. *Mitatut suoritusajat.*

Solujen määrä	Sivun kokonaislataus-aika (ms)	JavaScriptin suoritus-aika (ms)
10	1 180	820
100	1 280	909
10 000	2 290	1709

Mittaustuloksista nähdään, että sivun latausaika ja JavaScriptin suoritusajat kasvavat, kun sovelluksessa piirrettävän datan määrä kasvaa. Kuitenkin myös suurimmalla 10 000 solun datamäärällä sivun kokonaislatausaika ja JavaScript-suoritusaika pysyvät suhteellisen pieninä. Suurin testattu datamäärä ylittää normaalit käyttötilanteet, joita yrityksen sovelluksissa tulee. Pienemmillä 10 ja 100 solun datamäärillä mittausajat eivät merkittävästi eroa toisistaan. Tuloksissa on myös huomioitava, että testauslaitteiston suorituskyky oli varsin matala, eli esimerkiksi laitteiston muistimäärä oli vain 4 gigatavua ja laitteiston suoritin oli vuodelta 2010. Tehokkaammilla työasemilla suoritusajat voisivat olla huomattavasti nopeampia.

6.5 Tulosten arviointi ja johtopäätökset

Tutkimuksen tavoitteena oli selvittää valittujen tekniikoiden soveltumista yrityksen nykyisiin ja tuleviin tuotteisiin. Tutkimuksessa selvitettiin ensin kirjallisuudesta tekniikoiden eroja. Tekniikoiden vertailua varten selvitettiin yrityksen tarpeet tekniikoilta. Tavoitteena oli valita tarkempaan tutkimukseen tekniikka, joka kirjallisuuden perusteella olisi sopiva yrityksen nykyisiin tuotteisiin. Tämän jälkeen valituilla tekniikoilla toteutettiin sovellus, jolla saatiin tarkempaa tietoa tekniikoista, sekä niiden käyttämisestä web-sovelluksen toteutuksessa.

Tutkimuksessa toteutettiin React-kirjastoa käyttävä responsiivinen web-sovellus. Sovelluksen arviointiin määritettiin kriteerit ennen sovelluksen tekemistä. Taulukossa 7 on listattuna nämä tekniikan arvioinnissa käytetyt kriteerit, sekä miten kriteerin täyttymistä testattiin ja tutkimuksen tulos kyseisen kriteerin perusteella.

Taulukko 7. *Web-tekniikalle valitut arviointikriteerit, tutkimuksen tyyppi ja tutkimuksen tulos.*

Kriteeri	Mitä tutkimuksessa tehtiin	Tulos
Yhteensopivuus yrityksen nykyisiin kehitysympäristöihin. (Visual Studio)	Yrityksellä on käytössä Microsoft Visual Studio -kehitysympäristö. Demosovellus kehitettiin käyttäen Microsoft Visual Studio -kehitysympäristöllä. Lisäksi käytettiin Visual Studion paketinhallintaa ja testaustyökaluja.	React-sovelluksia voidaan kehittää SW-Development kehitystyökaluilla.
Tekniikan helppokäyttöisyys.	React oli kirjallisuudessa mainittu helppokäyttöiseksi. Lisäksi JSX:n muuttujien tiukka tyypitys on helpompi, kuin JavaScriptin dynaaminen tyypitys. Demosovelluksen perusteella React arvioitiin subjektiivisesti helppokäyttöiseksi.	React ja JSX -tekniikat ovat helppokäyttöisiä.
Yhteensopivuus modernien web-selainten kanssa.	Demosovellusta testattiin kolmella modernilla web-selaimella. Testatut selaimet olivat: Google Chrome versio 56.0.2924.76, Mozilla Firefox versio 52.0.4, Microsoft Edge versio. 38.14393.0.0.	Demosovellus toimi kaikilla testatuilla web-selaimilla. Tuloksena saatiin, että React on yhteensopiva modernien web-selainten kanssa.
Suorituskyky suurilla datamäärillä.	Demo sovelluksen suorituskykyä testattiin Chrome-selaimen kehittäjätyökaluilla. Testeissä suoritettiin sivulatauksia eri suuruisilla datamäärillä: 10, 100, 10000 alkiota.	Sovelluksen suorituskyky oli hyvä myös suurilla datamäärillä.
Soveltuminen responsiivisiin käyttöliittymiin	Tutkimuksessa toteutettiin responsiivinen web-sovellus, jota testattiin muuttamalla web-selaimen ikkunan kokoa. Lisäksi kirjallisuudesta selvitettiin, että React-kirjasto voidaan yhdistää muihin responsiivisiin käyttöliittymäkirjastoihin, kuten Bootstrapiin.	Toteutettu web-sovellus ja React-tekniikka soveltuvat responsiivisiin käyttöliittymiin.

Yhteenvetona tutkimuksen tuloksena saadaan, että React-kirjasto sopii hyvin yrityksen tuotteisiin. Yhtiön tuotekehityksessä on tavoitteena saada kaikista web-sovelluksista responsiivisia. React-sovelluksista on mahdollista tehdä responsiivisia. Lisäksi React-kirjaston tehokkuus on paras verratuista JavaScript-kirjastoista. React-kirjaston tehokkuus on mainittu kirjallisuudessa ja sama huomio tehtiin myös tämän työn suorituskykytesteissä.

React-kirjaston arkkitehtuuri eroaa muista yleisimmistä JavaScript-kirjastoista, kuten jQuerysta, joka on yrityksen tuotteissa käytössä. Tämän johdosta yrityksen nykyisien sovelluskehittäjien pitäisi myös opetalla uusi tapa tehdä SPA-sovelluksia, jos React otettaisiin käyttöön. Lisäksi työssä käytettiin JSX-ohjelmointikieltä, joka ei ole yrityksellä nykyisin käytössä. Opetteluun kuluva aika pitää ottaa huomioon, jos React halutaan ottaa käyttöön.

6.6 Jatkotutkimusaiheet

Tutkimuksessa selvitettiin Reactin käyttämistä melko suppeassa sovelluksessa. Sovelluksen jatkokehitykseen on useita mahdollisuuksia. Sovellus voitaisiin integroida suoraan yrityksen tuotteeseen ja tutkia kuinka suuren työmäärän integrointi vaatii. Tällöin voitaisiin ladata dataa yrityksen testitietokannoista ja verrata sovelluksen suorituskykyä nykyisin käytössä oleviin puurakenteisiin. Tällöin saataisiin tarkempia tuloksia suorituskyvyn parantumisesta Reactilla.

Työssä toteutettu sovellusta voitaisiin lisäksi kehittää useilla tavoin eteenpäin. Puurakenne voitaisiin yhdistää esimerkiksi gantt-suunnittelutaulukkoon. Tällöin puun yksitaso vastaisi yhtä tuotannon resurssia. Yrityksen tuotteissa Gantt-näkymässä esitetään suuria datamääriä. Lisäksi gantt-palikoiden välissä näytetään nuolilla niiden linkitys toisiin palikoihin. Suurilla datamäärillä tämä linkkien esittäminen nuolilla on aiheuttanut suorituskykyongelmia. Reactin tehokkuus voisi ratkaista nämä ongelmat, joten jatkokehityksessä tätä voitaisiin selvittää.

Tutkimuksessa toteutettua sovellusta voitaisiin myös käyttää pohjana muille toteutuksille. Puurakennetta voitaisiin muokata ja käyttää pelkistettynä listaelementtinä. Yrityksellä on käytössä useita listaratkaisuja, joihin Reactilla toteutettua listaa voitaisiin verrata. Esimerkiksi nykyisten listaelementtien suorituskykyongelmat voisivat olla Reactiin toteutuksella pienempiä.

7. YHTEENVETO

Tämän diplomityön tutkimuksen aiheena oli etsiä JavaScript-kirjasto, jota voitaisiin soveltaa SW-Development yrityksen tuotannonohjausjärjestelmän responsiivisessa web-sovelluksessa. Yrityksen järjestelmät (SWD^{PES}) ovat aikaisemmin perustuneet pääosin Windows-käyttöjärjestelmälle asennettaviin ratkaisuihin. Yrityksen tuotteissa ollaan kuitenkin siirtymässä web-teknologioihin ja pilvipalveluratkaisuihin. Silloin päästään alustariippumattomiin ratkaisuihin, jotka toimivat sekä mobiililaitteilla että täysikokoisilla tietokoneilla. Tavoitteena tässä työssä oli valita JavaScript-kirjastoista yrityksen web-sovelluksiin soveltuva käyttöliittymäkirjasto, sekä saada yritykselle tarkempaa tutkimustietoa front-end-tekniikoiden kehityksestä ja niiden soveltamisesta yrityksen tuotteisiin.

Työssä taustoitettiin ensin web-teknologioita yleisesti sekä esiteltiin, miten responsiiviset web-sovellukset eroavat vanhemmista staattisista web-sovelluksista (luku 2). Lisäksi luvussa 3 käytiin tarkemmin läpi responsiivisiä web-sovelluksia, jotka sopeutuvat erikokoisille näytöille, ilman erillisiä mobiilisivustoja. Luvussa 4 selvitettiin yrityksen SWD^{PES}-ohjelmiston nykyistä toteutusta ja tulevaisuuden kehityssuuntaa.

Seuraavaksi tässä diplomityössä tehtiin kvalitatiivinen tutkimus JavaScript-kirjastoista. Ensimmäisenä tutkittiin useiden JavaScript-kirjastojen ominaisuuksia. Vertailuun valittiin tämän hetken suosituimmat ja eniten käytetyt kirjastot. Lisäksi valittiin kriteerit, joilla JavaScript-kirjastoja arvioitiin. Kriteerien perusteella valittiin demosovelluksessa käytettävä JavaScript-kirjasto. Demosovelluksella tutkittiin, miten React-kirjasto soveltuu yrityksen tuotteisiin. Myös demosovellusta varten valittiin kriteerit, joilla sovellusta evaluoitiin. Evaluointikriteereihin kuului muun muassa suorituskyky ja helppokäyttöisyys.

Web-sovellukseen luotiin responsiivinen ja dynaamisesti muokattava puurakenne, jota voitiin käyttää web-selaimessa. Kyseisen sovelluksen toteutuksella tutkittiin miten tekniikka sopisi yrityksen PES-järjestelmän näkymien esittämiseen. Sovellusta ei integroitu yrityksen järjestelmään vielä tässä vaiheessa, mutta jatkokehityksessä tämä on mahdollista. Jatkokehityksessä se voitaisiin myös yhdistää yrityksen tietokantadataan. Sovelluksen kehittäminen oli suoraviivainen prosessi ja se antoi hyvän kuvan tekniikan soveltuvuudesta yritykselle. Sovelluksessa käytetyn React-kirjaston suurimpia etuja ovat helppokäyttöisyys ja hyvä suorituskyky. Lisäksi sen käyttäminen kehityksessä onnistuu yrityksen nykyisillä kehitystyökaluilla.

Tutkimuksen lopuksi toteutettua demosovellusta arvioitiin etukäteen valituilla kriteereillä. Tekniikan kriteerit valittiin yrityksen toiveiden mukaisesti. Näihin kriteereihin verrattaessa React-kirjasto sekä demosovellus sopivat yrityksen tuotteisiin. Yrityksellä on tuotekehityksessä web-teknologioihin perustuvia tuotannonohjausjärjestelmiä. Tämän

työn tutkimuksen perusteella annettiin suositus, että Reactia voi käyttää yrityksen tulevissa web-sovelluksissa ja että React-kirjastoa kannattaa verrata myös yrityksen nykyisiin ratkaisuihin.

Tutkimusta voitaisiin seuraavaksi jatkaa esimerkiksi integroimalla tuotettu sovellus nykyisiin järjestelmiin ja tietokantoihin. Tällöin voitaisiin saada tarkempia suorituskykytuloksia, sekä niille vertausarvoja nykyisistä web-ratkaisuista. Toinen vaihtoehto olisi käyttää demosovelluksen puurakennetta gantt-tuotannonsuunnittelunäkymässä, joka on laajassa käytössä yrityksen nykyisissä tuotannonohjausjärjestelmissä.

8. LÄHTEET

[1] StatCounter, **Mobile and tablet internet usage exceeds desktop for first time worldwide**, StatCounter, web page, 2016. Available (accessed 3.4.2017): http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide.

[2] P. Fraternali, G. Rossi, F. Sánchez-Figueroa, Rich internet applications, IEEE Internet Computing, Vol. 14, No. 3, 2010, pp. 9-12.

[3] I. Hickson, R. Berjon, S. Faulkner, T. Leithead, E. Doyle Navara, O'Connor Edward, S. Pfeiffer, **HTML5**, W3C, web page, 2014. Available (accessed 16.10.2016): https://www.w3.org/TR/html5/.

[4] W3Schools, HTML5 Introduction, W3Schools, web page, 2016. Available (accessed 1.11.2016): https://www.w3schools.com/html/html5_intro.asp.

[5] W3Schools, JavaScript Tutorial, W3schools, web page, 2016. Available (accessed 16.10.2016): https://www.w3schools.com/js/.

[6] Mozilla Developer Network, **JavaScript technologies overview**, Mozilla Foundation, web page, 2016. Available (accessed 1.11.2016): https://developer.mozilla.org/en-US/docs/Web/JavaScript/JavaScript_technologies_overview.

[7] D. Flanagan, JavaScript: The Definitive Guide, 4th ed. O'Reilly Media, 2004, 2-461 p.

[8] FastWebStart, **A Modern HTML Introductory Tutorial**, Fast web start, web page, 2016. Available (accessed 31.10.2016): http://fastwebstart.com/modern-html-tutorial/.

[9] Mozilla Developer Network, **A re-introduction to JavaScript (JS tutorial)**, Mozilla developer network, web page, 2016. Available (accessed 30.9.2016): https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript.

[10] W3C, ***Cascading Style Sheets* home page**, W3C, web page, 2016. Available (accessed 1.11.2016): https://www.w3.org/Style/CSS/.

[11] W3Schools, **JavaScript HTML DOM**, w3schools.com, web page, 2016. Available (accessed 1.11.2016): https://www.w3schools.com/js/js_htmldom.asp

[12] M. Takada, **Single page apps in depth**, 1 st ed. http://singlepageappbook.com, http://singlepageappbook.com, 2015, 59-59 p.

[13] G. Fink, I. Flatow, **Pro Single Page Application Development: Using Backbone.js and ASP.NET**, 1st ed. Apress, 2014, 307 p.

[14] W3Schools, AJAX Introduction, w3schools.com, web page, 2016. Available (accessed 1.11.2016): http://www.w3schools.com/xml/ajax_intro.asp.

[15] J. Farrell, G.S. Nezlek, Rich internet applications the next stage of application development, 2007 29th International Conference on Information Technology Interfaces, IEEE, pp. 413-418.

[16] Quintagroup, **Responsive web design - adapt, respond and overcome**, Quintagroup, web page, 2016. Available (accessed 1.11.2016): http://quintagroup.com/services/web-design/responsive-web-design.

[17] M.H. Baturay, M. Birtane, Responsive web design: a new type of design for web-based instructional content, Procedia-Social and Behavioral Sciences, Vol. 106, 2013, pp. 2275-2279.

[18] W3Schools, **HTML Responsive Web Design**, www.w3schools.com, web page, 2016. Available (accessed 1.11.2016): https://www.w3schools.com/html/html_responsive.asp.

[19] E. Marcotte, Responsive Web Design, A Book Apart (2011), 2011.

[20] W3C, Media Queries, W3C, web page, 2016. Available (accessed 1.12.2016): https://www.w3.org/TR/css3-mediaqueries/.

[21] W3Schools, **CSS3 Browser Support Reference**, w3schools.com, web page. Available (accessed 1.10.2016): https://www.w3schools.com/cssref/css3_browsersupport.asp.

[22] C. Buckler, **Browser Trends January 2016: 12 Month Review**, Sitepoint, web page. Available (accessed 15.10.2016): https://www.sitepoint.com/browser-trends-january-2016-12-month-review

[23] SW-Development, SW-Development, Sisäinen dokumentaatio, SW-Development OY, web page, 2016. Available (accessed 1.9.2016): www.swd.fi

[24] H. Stadtler, C. Kilger, Supply chain management and advanced planning, Concepts, Models, Software and Case Studies, Vol. 4, 2008

[25] H. Wang, L. Liu, Y. Fei, T. Liu, A collaborative manufacturing execution system oriented to discrete manufacturing enterprises, Concurrent Engineering, Vol. 24, No. 4, 2016, pp. 330-343.

[26] O. Thompson, Does MES contribute to ERP success? Food Engineering, Vol. 78, No. 10, 2006, pp. 119

[27] E. Oinas, Tuotannonsuunnittelujärjestelmän suorituskykymonitorointi, Master's Thesis, Tampere University of Technology, 2015, 45 pp.

[28] Siemens Product Lifecycle Management Software, **Manufacturing Shop Floor Control**, Siemens, web page, 2016. Available (accessed 1.11.2016): http://camstar.industrysoftware.automation.siemens.com/en/resources/glossary/shop-floor-control/.

[29] J. Peltola, Tuotannonohjausjärjestelmän automaatiorajapinnan suunnittelu, Master's thesis, Tampere University of Technology, 2014, 1 p.

[30] ECMA international (ed.). 2006. Standard ECMA-334, C# Language Specification. 4th ed. ECMA International. 553 p.

[31] Microsoft, .NET Platform Guide, Microsoft, web page, 2016. Available (accessed 1.12.2016): https://docs.microsoft.com/fi-fi/dotnet/articles/standard/index.

[32] Github, GitHub, web page. Available (accessed 12/15/2016): www.github.com.

[33] W3Schools, **Bootstrap 3 Tutorial**, W3schools.com, web page, 2016. Available (accessed 1.1.2017): https://www.w3schools.com/bootstrap/

[34] Google, **Angular Docs,** TypeScript, Google, web page, 2017. Available (accessed 27.1.2017): https://angular.io/docs/ts/latest.

[35] Google, **Architecture Overview**, Google, web page, 2017. Available (accessed 27.1.2017): https://angular.io/docs/ts/latest/guide/architecture.html.

[36] W3Techs, **Usage of JavaScript libraries for websites**, w3techs.com, web page, 2017. Available (accessed 1.2.2017): https://w3techs.com/technologies/overview/javascript_library/all

[37] The jQuery Foundation, **jQuery Learning Center**, The jQuery Foundation, web page, 2017. Available (accessed 1.3.2017): https://learn.jquery.com/.

[38] C. Buckler, **Do You Really Need jQuery?**, Sitepoint.com, web page, 2013. Available (accessed 12.4.2017): https://www.sitepoint.com/do-you-really-need-jquery/.

[39] A. De Rosa, J. Likness, J. Looper, N. Anderson, T. Motto, T. Vantoll, **Is jQuery Still Relevant?**, Telerik Developer Network, web page. Available (accessed 15.4.2017): http://developer.telerik.com/featured/is-jquery-still-relevant/.

[40] Facebook Inc., **React,** A JavaScript library for building user interfaces, Facebook Open Source, web page, 2017. Available (accessed 1.2.2017): https://facebook.github.io/react/.

[41] C. McKeachie, **React.js and How Does It Fit In With Everything Else?**, www.funnyant.com, web page, 2014. Available (accessed 1.2.2017): http://www.funnyant.com/reactjs-what-is-it/.

[42] Facebook Inc., **ReactJS.NET**, Facebook Inc., web page, 2017. Available (accessed 1.2.2017): https://reactjs.net/.

[43] GrapeCity, ComponentOne, GrapeCity, web page, 2017. Available (accessed 14.2.2017): www.componentone.com.

[44] J. Hannah, **Choosing the Right JavaScript Framework for the Job**, Lullabot, web page, 2015. Available (accessed 2.3.2017): https://www.lullabot.com/articles/choosing-the-right-javascript-framework-for-the-job.

[45] S. Peyrott, **More Benchmarks: Virtual DOM vs Angular 1 & 2 vs Others**, auth0.com, web page, 2016. Available (accessed 20.2.2017): https://auth0.com/blog/updated-and-improved-more-benchmarks-virtual-dom-vs-angular-12-vs-mithril-js-vs-the-rest.

[46] J. Touffe-Blin, **Angular Chart**, Github.com, web page, 2017. Available (accessed 1.3.2017): http://jtblin.github.io/angular-chart.js/.

[47] Npmjs, **These are the docs you're looking for**, NPM, web page, 2017. Available (accessed 1.3.2017): https://docs.npmjs.com/.

[48] Codementor, Muhammad, **React vs Angular 2: Comparison Guide for Beginners**, Codementor Community, web page, 2016. Available (accessed 10.3.2017): https://www.codementor.io/codementorteam/react-vs-angular-2-comparison-beginners-guide-lvz5710ha.