Olli-Pekka Kaukola

ReactJS web-käyttöliittymän toteutuksessa

Opinnäytetyö
Kevät 2018
SeAMK Tekniikka
Tieto- ja viestintätekniikka

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Olli-Pekka Kaukola

Työn nimi: ReactJS web-käyttöliittymän toteutuksessa

Ohjaaja: Hilkka Niemelä

Vuosi: 2018 Sivumäärä: 27 Liitteiden lukumäärä: 0

Insinöörityössä tutustuttiin React JavaScript-kirjastoon. Työn tavoitteena oli suunnitella ja toteuttaa web-käyttöliittymä ReactJS-kirjastoa käyttäen.

Työn teoriaosuudessa syvennyttiin käyttöliittymän suunnitteluun ja ReactJS-kirjaston toimintaperiaatteisiin. Käytännön osuudessa suunniteltiin ja toteutettiin web-käyttöäliittymä käyttäen ReactJS-kirjastoa.

Lopputuloksena syntyi tavoitteiden mukainen selkeä ja helppokäyttöinen käyttöliittymä. Käyttöliittymällä voidaan tallentaa tiedot käyttäjän omistamista elokuvista.

Avainsanat: ReactJS, JavaScript, käyttöliittymä

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Engineering

Author: Olli-Pekka Kaukola

Title of thesis: ReactJS Web User Interface Implementation

Supervisor: Hilkka Niemelä

Year: 2018 Number of pages: 27

In this thesis the aim was to get familiar with the React JavaScript Library. The second aim of the thesis was to design and implement a web user interface using the ReactJS library.

The theoretical part of the thesis gave a deeper insight into the designing of user interfaces and into the ReactJS library's operating principles. In the practical part, a web interface was designed and implemented using the ReactJS library.

The final result was a clear and easy-to-use user interface. The user interface can be used to store information about the writer’s own movie collection.

Keywords: ReactJS, JavaScript, user interface

SISÄLTÖ

Kuvaluettelo

Käytetyt termit ja lyhenteet

API	Application Programming Interface eli ohjelmointirajapinta, jonka mukaan ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja keskenään.
DOM	Document Object Model eli dokumenttioliomalli on tapa kuvata rakenteinen dokumentti.
MongoDB	Ilmainen, avoimen lähdekoodin tietokantaohjelma.
ReactJS	Facebookin kehittämä JavaScript-kirjasto käyttöliittymien rakentamiseen.
UI	User Interface eli käyttöliittymä on osa tuotetta, minkä kautta käyttäjä käyttää tuotetta.
Virtual DOM	Kopio DOM-mallista, jonka tarkoituksena on verrata muutoksia alkuperäiseen DOM-malliin.

1 JOHDANTO

1.1 Työn tausta

ReactJS on suosituimpia JavaScript-sovelluskehyksiä. Sovelluskehykset nopeuttavat ohjelmistojen kehitystyötä tuomalla käytettäväksi valmiita toiminnollisuuksia, tämä vähentää kirjoitettavan koodin määrä.

Opinnäytetyön sovelluskehykseksi valikoitui ReactJS, koska se on monipuolinen ja suosiotaan koko ajan kasvattava sovelluskehys. Kirjoittajalla on vähän kokemusta kyseisestä sovelluskehyksestä ennestään ja tarkoitus on perehtyä kyseiseen kirjastoon enemmän.

1.2 Työn tavoite

Työn tavoitteena on tutustua ja esitellä React JavaScript-kirjasto sekä suunnitella ja ohjelmoida helppokäyttöinen web-käyttöliittymä käyttäen ReactJS-kirjastoa.

Työstä pitäisi selvitä lukijalle, mitkä ovat hyvän käyttöliittymän perusteet ja mitenkä käyttöliittymä luodaan käyttäen ReactJS-kirjastoa.

1.3 Työn rakenne

Luvussa 2 esitellään käytettävyysstandardeja sekä hyvän käyttöliittymän lähtökohdat. Luvussa 3 käydään läpi ReactJS-kirjastoa. Luvussa 4 käydään läpi insinöörityössä toteutettua käyttöliittymää sen toimintaa ja rakennetta, visuaalista ilmettä sekä jatkokehitysideoita. Viimeisessä luvussa on yhteenveto.

2 KÄYTTÖLIITTYMÄN SUUNNITTELU

2.1 Mitä on käyttöliittymäsuunnittelu

Käyttöliittymäsuunnittelu (User Interface Design), johon usein viitataan lyhenteellä UID, keskittyy siihen, miten valmiin tuotteen tai palvelun käyttämisestä tehdään helppoa eri käyttäjille. Käyttöliittymäsuunnittelussa määritellään mm. käyttäjille tarpeelliset toiminnot ja elementit sekä käyttöliittymän rakenne. Suunnittelussa korostuu käyttäjien tarpeiden sekä käyttöympäristöjen (esim. mobiililaitteet) ymmärtäminen. (Muranen & Harmainen [Viitattu 26.3.2018])

Käyttöliittymäsuunnittelussa ennakoidaan, mitä käyttäjät aikovat tehdä sekä varmistutaan siitä, että käyttöliittymän elementit ja interaktio ovat ymmärrettäviä, helposti saavutettavissa ja vaivattomia käyttää. Käyttöliittymä yhdistää ja konkretisoi interaktiosuunnittelun, visuaalisen suunnittelun ja informaatioarkkitehtuurin. (Nieminen [Viitattu 26.3.2018]).

2.2 Käytettävyysstandardit ja ohjeistukset

Seuraavassa käsitellään käytettävyysstandardeja. Käyttöliittymän suunnittelun tueksi on kehitetty standardeja. Seuraavassa esitellään tärkeimmät käytettävyyteen liittyvät ISO-järjestelmän standardit.

2.2.1 ISO 9241-151:2008

Standardi ISO 9241-151:2008, antaa ohjeita ohjelmistojen web-käyttöliittymien ihmiskeskeisestä suunnittelusta käytettävyyden lisäämiseksi. Web-käyttöliittymät koskevat joko kaikkia Internetin käyttäjiä tai suljettuja käyttäjäryhmiä, kuten organisaation jäseniä, asiakkaita ja / tai yrityksen tai muiden erityisten käyttäjäyhteisöjen toimittajia. Tässä keskitytään seuraaviin standardissa ISO 9241:2008 annettuihin suosituksiin koskien wed-käyttöliittymän suunnittelua: korkean tason suunnittelupäätöksiin ja suunnittelustrategiaan, sisällön suunnitteluun, navigointiin ja hakuun

sekä sisällön esittelyyn. Eri tyyppisten käyttäjäagenttien käyttöliittymiä, kuten web-selaimia tai muita työkaluja, kuten web-kirjoitustyökaluja, ei käsitellä suoraan tässä osassa ISO 9241: 2008 (vaikka osa sen ohjeista voisi koskea myös näitä järjestelmiä). Web-käyttöliittymät esitetään henkilökohtaisessa tietokonejärjestelmässä, matka-viestinjärjestelmässä tai jollakin muulla verkkoliitännällä varustetulla laitteella. Vaikka ISO 9241: 2008: n tässä osassa annetut suositukset koskevat monenlaisia käytettävissä olevia ensisijaisia tekniikoita, mobiililaitteiden web-rajapintojen tai älylaitteiden suunnittelu voi edellyttää lisäohjeita, jotka eivät kuulu sen soveltamisalaan. Se ei anna yksityiskohtaista ohjeistoa teknisestä toteutuksesta eikä esteettisestä tai taiteellisesta suunnittelusta. (ISO 9241-151, 2008.)

2.2.2 ISO 9241-11

ISO 9242-11 -standardi (joka on osa ISO 9241 -sarjaa) antaa seuraavan määritelmän käytettävyydelle:

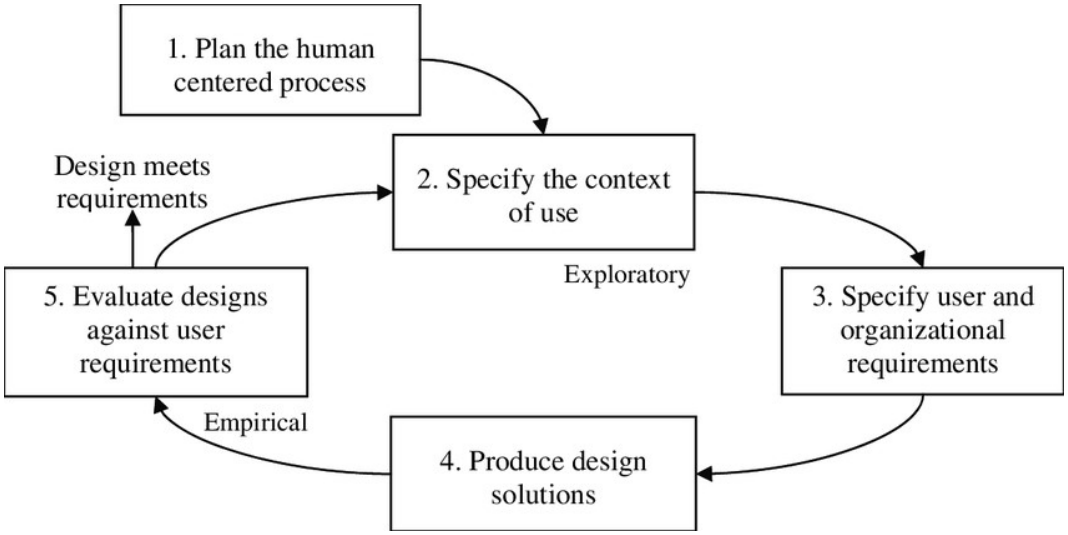
Käytettävyys tarkoittaa tarkoituksenmukaisuutta, tehokkuutta ja tyytyväisyyttä, jolla tuotteen määritellyt käyttäjät saavuttavat määritellyt tavoitteet tietyssä käyttöympäristössä. Standardi ISO 9241-11 selittää miten tunnistaa tiedon, joita on tarpeen ottaa huomioon määritettäessä tai arvioitaessa käytettävyyttä käyttäjien suorituskyvyn ja tyytyväisyyden mittaamiseksi. Standardi antaa ohjeita siitä, kuinka tuotteen käyttötapa ja käyttökelpoisuus ovat selkeästi kuvaillut. Se sisältää selityksen siitä, kuinka tuotteen käytettävyys voidaan määritellä ja arvioida osana laatujärjestelmää, esimerkiksi ISO 9001 -standardin mukaista laatujärjestelmää. Se selittää myös, miten käyttäjän toiminnan ja tyytyväisyyden mittauksia voidaan käyttää mittaamaan kuinka jokin tietojärjestelmän osa vaikuttaa koko käytössä ole-van tietojärjestelmän laatuun. (International standards for HCI and usability [Viitattu 26.3.2018].)

2.2.3 ISO 13407

ISO 13407 -standardi antaa ohjeita ihmiskeskeisestä suunnittelutoiminnasta koko interaktiivisten tietokonepohjaisten järjestelmien elinkaaren aikana. Se on työkalu suunnitteluprosesseja hallinnoiville käyttäjille ja ohjaa tietolähteitä ja standardeja,

jotka liittyvät ihmiskeskeiseen lähestymistapaan. Se kuvaa ihmiskeskeistä muotoilua monitieteisenä toimintana, johon sisältyy inhimillisiä tekijöitä ja ergonomiaosaamista ja -tekniikoita. Tavoitteena on parantaa vaikuttavuutta ja tehokkuutta, parantaa ihmisten työolosuhteita ja torjua käytöstä johtuvia mahdollisia haittavaikutuksia ihmisten terveyteen, turvallisuuteen ja suorituskykyyn. (International standards for HCI and usability [Viitattu 26.3.2018].)

Suositeltava prosessi on esitetty kuvassa 1



Kuva 1. Suositeltava prosessi ihmiskeskeisessä suunnittelussa. (ResearchGate [Viitattu 26.3.2018].)

2.3 Hyvän käyttöliittymän lähtökohdat

Standardit eivät anna suoraan ohjeistusta hyvän käyttöliittymän suunnitteluun. Niihin perustuen on kehitetty erilaisia ohjeita, joita noudattamalla päästään hyvään lopputulokseen.

Käyttöliittymän suunnittelussa tulisi muistaa huomioida monia asioita ja miettiä, kuinka käyttökokemusta voidaan parantaa käyttäjän näkökulmasta. Seuraavassa listassa on kymmenen muistisääntöä, jotka kannattaa ottaa huomioon käyttöliittymää suunniteltaessa. Kannattaa kuitenkin muistaa aina, että parhaat käyttöliittymät ovat sellaisia, joihin käyttäjä ei kiinnitä huomiota niitä käyttäessään. Jokaisen muistisäännön kohdalla paneudutaan itse asiaan enemmän. (Graafinen 2015.)

Tunne käyttöliittymän käyttäjä. Ensimmäinen muistisääntö käyttöliittymää suunniteltaessa on, että pitää tuntea käyttäjä, joka käyttöliittymää tulee myöhemmin käyttämään. Huomioitavia kohtia tässä on esimerkiksi, minkä ikäinen hän on, millaisiin asioihin käyttäjä on tottunut, minkälaisia aikaisempia kokemuksia hänellä on jo käyttöliittymistä ja minkälaisia taitoja hänellä saattaa olla. Käyttöliittymää ei missään nimessä kannata lähteä suunnittelemaan valitsevien trendien mukaisesti, sillä saattaa olla, että ne eivät sovi käsiteltävään tapaukseen. Käyttöliittymän tekijän täytyy tunnistaa, mitä käyttäjä haluaa saada aikaiseksi ja auttaa häntä saavuttamaan tavoitteensa mahdollisimman yksinkertaisesti ja helposti. (Graafinen 2015.)

Kiinnitä huomiota asioihin, joita on aiemmin jo opittu. Käyttöliittymästä tulee tehdä sellainen, että se noudattaa aikaisemmissa käyttöliittymissä toimivia sääntöjä ja periaatteita eli sellainen, mihin käyttäjä on jo tottunut. Ei ole järkevää lähteä keksimään lamppua uudelleen ja opettaa käyttäjälle täysin uudenlaisia tapoja toimia, jos niille on jo löydetty toimiva ja hyväksi havaittu tapa tehdä asioita. Käyttäjillä on yleensä oma totuttu tapa tehdä asioita. Käyttäjän tulee tuntea olonsa kotoisaksi käyttöliittymää käyttäessään. Hyvä ja toimiva tapa on selata internetissä jo valmiiksi suosittuja sivustoja ja kokeilla erilaisia sovelluksia ja huomioida, kuinka muut ovat ratkaisseet mahdollisia ongelmakohtia. Näin ollen on helppo poimia parhaat neuvot ja ideat ja soveltaa niitä käytännössä. (Graafinen 2015.)

Ole johdonmukainen. Hyvässä käyttöliittymässä käyttäjä pystyy ennakoimaan käyttöliittymässä olevia toimintaperiaatteita, esimerkiksi erilaisten valikoiden ja valintaruutujen tulisi pysyä samannäköisinä ja toimia samalla tavalla käyttöliittymän joka kohdassa. Painikkeiden ja vierityspalkkien tulee myös näyttää samalta ja toimia samalla tavalla kaikkialla. Esimerkiksi, jos jotain painiketta tulee painaa vain kerran, tulee silloin kaikissa samanlaisissa painikkeissa olla sama toimintaperiaate ja logiikka. Toisella tavalla toimivien painikkeiden ja palkkien tulee näyttää käyttäjälle tarpeeksi erilaisilta, jotta tämä pystyy hahmottamaan eron näiden välillä tarpeeksi selkeästi. Käyttöliittymän hyvä visuaalinen ilme mahdollistaa sen, että käyttäjä saa tietoa, miten mikäkin toimii, mikä kaikkien painikkeiden ja palkkien tarkoitus käyttöliittymässä on. Erilaiset värit ja symbolit painikkeissa kertovat käyttäjälle, mitä painikkeesta tapahtuu. Ne auttavat hahmottamaan käyttöliittymän toimintaa. Lisäksi

väreillä ja symboleilla saadaan käyttöliittymä erottumaan muiden joukosta ja tehdään siitä kiinnostava. (Graafinen 2015.)

Luo hyvä visuaalinen ilme. Käyttöliittymä tulisi suunnitella graafisesti niin, että kaikki tärkeimmät ja kiinnostavimmat asiat nousevat esiin aivan ensimmäisinä ja saavat käyttöliittymästä isoimman roolin. Seuraavaksi järjestyksessä tulevat toisiksi tärkeimmät asiat saavat jo yleensä pienemmän rooli ja niin ketju jatkuu eteenpäin. Käyttöliittymässä kaikkien käytössä olevien eri elementtien väri, koko ja sijoittelu täytyy olla joka vaiheessa harkittua, ja kokonaisuuden on toimittava niin visuaalisessa kuin loogisessakin mielessä. Suunnitteluvaiheessa tulee välttää liian monimutkaisia ratkaisuja ja tulee luoda selkeät painotukset asioiden välille. (Graafinen 2015.)

Anna tarpeeksi palautetta. Oikein toimivan käyttöliittymän tulisi antaa palautetta käyttäjälleen. Käyttöliittymän tulisi kertoa käyttäjälle, milloin hän toimii oikein sekä milloin hän toimii väärin. Tämä voi näkyä käyttöliittymässä visuaalisena ohjeena, kuten punaisena huutomerkkinä puuttuvista tiedoista tai ruutuun avautuvana kiitosviestinä onnistuneesti annetusta asiakaspalautteesta. Suuri virhe on jättää käyttäjä epätietoiseen tilaan, onnistuiko hän siinä mitä hän yritti tehdä. Palautteen antaminen käyttäjälle antaa hyvän käyttäjäkokemuksen, mikä on äärimmäisen tärkeä asia. Tämä asia jää monelta huomioimatta, eikä sille anneta tarpeeksi painoarvoa käyttöliittymän suunnitteluvaiheessa. (Graafinen 2015.)

Anna virheet anteeksi. Vaikka käyttöliittymästä yritetään tehdä helppo, selkeä ja johdonmukainen, niin silti kaikki ihmiset tekevät joskus virheitä. Hyvän käyttöliittymän tulisikin pystyä antamaan pienet virheet anteeksi ja käyttäjälle mahdollisuus korjata tai muuttaa niitä. Esimerkiksi, kun käyttäjä täyttää jonkinlaista lomaketta, käyttäjälle tulisi suoda mahdollisuus muuttaa helposti jo syöttämiään tietoja ja pyytää käyttäjää tarkistamaan aiemmin lomakkeelle syöttämänsä tiedot ennen lähettämistä. Jokainen tietää, että mikään ei ole raivostuttavampaa, kuin aloittaa jokin asia täysin alusta ainoastaan sen takia, että käyttöliittymä ei mahdollista syötettyjen tietojen korjausta. (Graafinen 2015.)

Jos jokin muunlainen virhe aiheutuu käyttäjän omasta osaamattomuudesta tai pelkästä huolimattomuudesta, on silloin virheilmoituksen yhteydessä hyvä paikka opettaa käyttäjälle, kuinka samankaltaiset virheet ovat mahdollista jatkossa välttää (Graafinen 2015).

Valta käyttäjälle. Jossain vaiheessa käyttäjä tottuu ja oppii, miten käyttöliittymä toimii eri tilanteissa. Tällöin käyttäjälle tulee antaa mahdollisuuksia tehdä asiat hieman vaivattomammin. Näppäinkomennoilla tai erilaisilla oikoreiteillä voidaan helpottaa toistuvia asioita. Kun käyttäjän omat valintamahdollisuudet lisääntyvät, oppii hän käyttämään käyttöliittymää vielä vaivattomammin. Tämä luo positiivisen käyttäjäkokemuksen. (Graafinen 2015.)

Puhu käyttäjän kanssa samaa kieltä. Jokainen suunnittelija varmasti ajattelee, että hyvässä käyttöliittymässä jokainen pikseli, ikoni, symboli ja jopa kirjasintyyppi vaikuttaa käyttäjän kokemaan käyttäjäkokemukseen. Näin se yleensä onkin, mutta silloin se myös tarkoittaa sitä, että kaikki käyttöliittymässä näkyvä teksti vaikuttaa käyttäjän kokemukseen. Hyvät käyttöliittymät tarvitsevat myös hyvää ja oikeanlaista tekstiä toimiakseen käyttäjän vaatimalla tavalla. Tekstin tulee olla yksinkertaista, ei liian hienoja sanoja, joita käyttäjä ei ymmärrä. Lisäksi tekstin tulee olla johdonmukaista ja käyttäjän tarpeet täyttävää eli käyttäjälle hyödyllistä. Käyttöliittymässä näkyvä teksti luo käyttöliittymälle äänen. (Graafinen 2015.)

Yksinkertaista asiat. Asioiden yksinkertaistaminen on monesti käyttöliittymien suunnitteluissa vaikein kohta, mutta se on myös palkitsevin kohta. Yleensä kaikki asiat on mahdollista tehdä paljon yksinkertaisemmin ja helpommin, mutta se vaatii käyttöliittymän suunnittelijalta paljon ajattelua ja asioiden kyseenalaistamista. Suunnittelijan tulisi usein kysyä itseltään ”Tarvitseeko käyttäjä välttämättä juuri tätä painiketta? Voisiko tämän tekstin korvata vaikka kuvalla? Arvostaako käyttäjä juuri näitä pieniä asioita?” On hyvä muistaa, että yleensä parhaimmat käyttöliittymät ovat juuri niitä yksinkertaisimpia ja huomaamattomia käyttää. Hyvän käyttöliittymän tulisi pysytellä taustalla, sen ei tulisi varastaa koko esitystä. (Graafinen 2015.)

Kehittäminen jatkuu koko ajan. Parhaat käyttöliittymät ja tulokset syntyvät useimmiten erehdysten ja kokeiluiden ansiosta. Kannattaa testata, mikä todellakin toimii ja mikä ei toimi ollenkaan. Kaikki suunnittelijat tekevät jossain vaiheessa virheitä,

mutta parhaat ottavat niistä opikseen ja tekevät seuraavalla kerralla paremmin. Käyttäjältä saatu tieto ja palaute on parasta käyttöliittymän kehityksen kannalta. Palautteista saadaan tietoa käyttöliittymän toimivuudesta käytännössä. Tämä on tärkeä asia käyttöliittymää suunniteltaessa. Käyttäjät ovat se asiakaskunta, jolle suunnittelija käyttöliittymän tekee. Käyttäjien mielipiteitä tulee aina kunnioittaa ja heidän palautettaan kannattaa kuunnella. (Graafinen 2015.)

3 REACTJS

3.1 Deklaratiivinen ohjelmointimalli

React on JavaScript-työkalu, jonka avulla on helppo ymmärtää, rakentaa ja ylläpitää tilallisia ja tilattomia käyttöliittymiä. Se tarjoaa keinot määritellä ja jakaa käyttöliittymä UI-komponentteihin (tunnetaan myös React-komponentteina) käyttämällä HTML-tyylisiä solmuja, joita kutsutaan React-solmuiksi. React-solmut muuttuvat lopulta UI-mallinnuksen muotoisiksi (esim. HTML / DOM, canvas, svg, jne.). (React Enlightment [Viitattu 26.3.2018].)

ReactJS-kehys ei ole täydellinen. Tarkoituksena on pikemminkin käyttää tekniikoita, jotka on järkeviä sovellusten erilaisten ongelmien suhteen, ja käyttää ReactJS-työkaluja katseluun ja käyttöliittymäkehitykseen. (Hayward 2015, 5.)

React käyttää deklaratiivista ohjelmointia eli selittävän ohjelmoinnin ajatusmallia. Se on yksi niistä syistä, mikä tekee Reactin käytöstä tehokasta. Reactin käyttämisen kannalta on tärkeää ymmärtää, mitä deklaratiivinen ohjelmointi tarkoittaa, ja mikä on sen eroavaisuus imperatiiviseen ohjelmointiin. Imperatiivinen ohjelmointi kuvastaa kuinka asiat toimii vaihe vaiheelta, deklaratiivinen ohjelmointi kertoo, mitä halutaan saavuttaa. (Bertoli 2017, 8.)

3.2 Virtual DOM

Virtuaalinen DOM (VDOM) on ohjelmointikonsepti, jossa käyttöliittymän ihanteellinen tai "virtuaalinen" esitys pidetään muistissa ja se synkronoidaan "todellisen" DOM-mallin kanssa kirjastossa. Tätä prosessia kutsutaan sovinnoksi. Tämä lähestymistapa mahdollistaa Reactin deklaratiivisen API:n. Voidaan sanoa Reactille, mitä tilaa halutaan käyttää käyttöliittymässä, ja se varmistaa, että DOM vastaa kyseistä tilaa. Tämä tiivistää attribuutin manipuloinnin, tapahtumien käsittelyn ja manuaalisen DOM-päivityksen, joita muuten täytyisi käyttää sovelluksen rakentamisessa. Virtuaalinen DOM on enemmän malli kuin tietty teknologia. React-maailmassa termi virtuaalinen DOM liittyy yleensä React-elementteihin, koska ne ovat käyttöliittymää

edustavia objekteja. React kuitenkin myös käyttää sisäisiä objekteja, joita kutsutaan kuiduiksi, pitämään lisätietoa komponenttipuusta. Niitä voidaan myös pitää osana virtuaalisen DOM-mallin toteutusta Reactissa. Tämä toimintatapa tekee Reactista tehokkaan. (React [Viitattu 26.3.2018].)

3.3 Komponentit

React on komponenttipohjainen, mikä tarkoittaa sitä, että komponenttien avulla voidaan jakaa käyttöliittymän itsenäisiin, uudelleenkäytettäviin palasiin ja miettiä jokaista palasta erikseen. Komponentit voidaan asettaa muihin komponentteihin, kuten työssä on esimerkiksi navigointipalkin kohdalla tehty kuvassa 6. Näin voidaan käyttää samaa komponentin abstraktiota mille tahansa yksityiskohtaiselle tasolle. React-sovelluksessa ilmaistaan painikkeet, lomakkeet, valintaikkunat ja näyttö yleensä osina. (React [Viitattu 3.4.2018].)

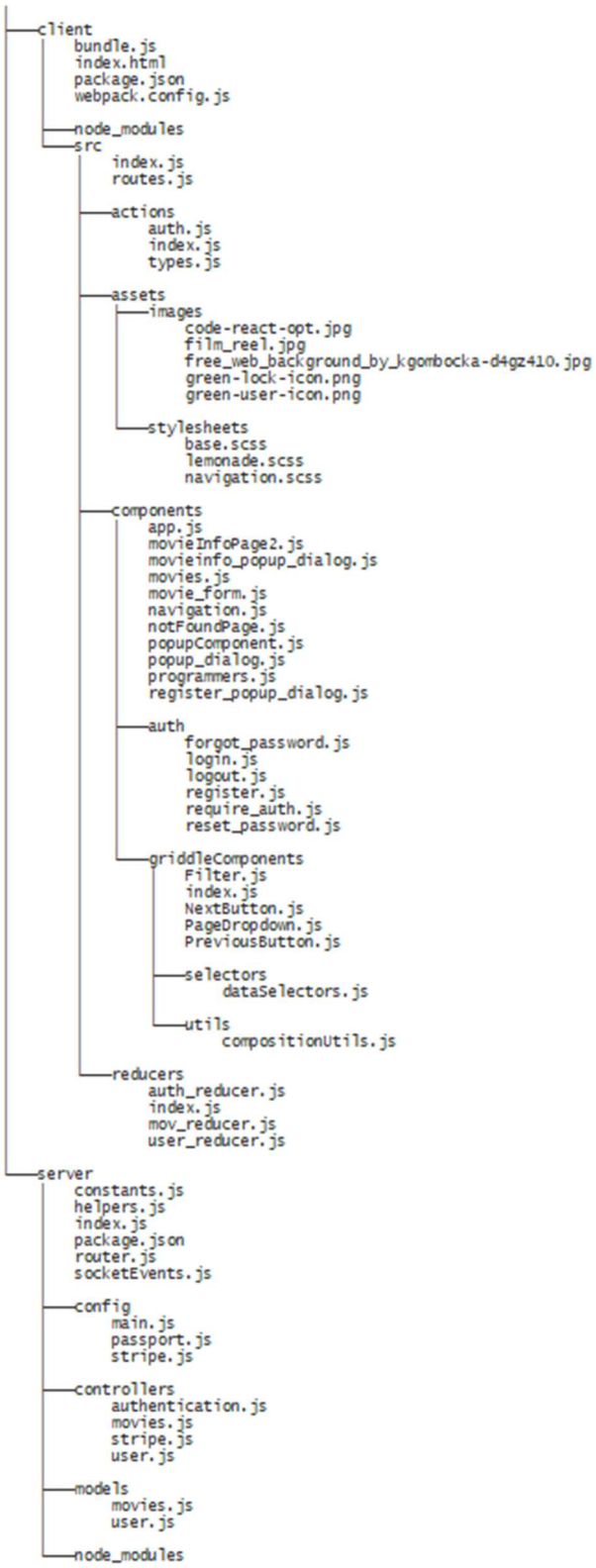
Tämä vähentää huomattavasti kirjoitettavan koodin määrää, kun ei tarvitse esimerkiksi koodata toiminnallisesti samanlaista painiketta uudestaan, vaan käytetään kertaalleen luotua uudestaan.

4 KÄYTTÖLIITTYMÄN TOTEUTUS

Käyttöliittymän ohjelmointiin käytettiin Microsoftin ilmaista Visual Studio Codea ja siinä Node.js JavaScript tulkkia sekä npm-paketinhallintaa. Ohjelma suunniteltiin elokuvaharrastajille, joilla on laaja elokuvakokoelma, helpottamaan kokoelman hallintaa. Hyvän käyttöliittymän lähtökohtien mukaisesti käyttöliittymä ohjelmoitiin hyvin yksinkertaiseksi. Sen käyttöä ei tarvitse kummemmin ajatella, eikä erillisiä ohjeita käyttöön tarvita. Ohjelmoinnin ja testauksen aikana karsittiin turhia toimintoja pois. Käyttöliittymässä on vain elokuvien lisäys ja poisto sekä käyttäjän lisäys. Tietokantana käytettiin avoimen lähdekoodin MongoDB-tietokantaa.

4.1 Sovelluksen rakenne ja toiminta

Kun oli suunniteltu mitä käyttöliittymässä tarvitaan, oli ohjelmointi helppoa aloittaa. Ohjelmointi aloitettiin luomalla hakemistorakenne. Työssä käytettiin kuvan 2 kaltaista rakennetta helpottamaan hakemista. Laajemmassa projektissa tämänlainen rakenne parantaa työtehoa. Projektin palvelin (server) ja pääteohjelma (client) sijoitettiin omiin hakemistoihinsa. Pääteohjelman puolella kuvat (images), komponentit (components) ja tyylitiedostot (stylesheets) sijoitettiin omiin hakemistoihinsa ja palvelimen puolella asetukset (config), kontrollerit (controllers) ja mallit (models) omiin hakemistoihinsa. Node_modules-hakemistot sisältävät projektissa käytetyt paketit, jotka npm-paketinhallinta luo paketteja asennettaessa.



Kuva 2. Ohjelmiston hakemistorakenne

Kun hakemistorakenne oli luotu, luotiin package.json-tiedosto komennolla npm init (kuva 3), että voitiin asentaa projektissa käytetyt paketit. Pakettien asennus tapahtuu komennolla npm install paketin nimi -save eli esimerkiksi React-paketin asennus tapahtuu komennolla npm install react –save (kuva 4).

```
reactApp\client>npm init
```

Kuva 3. Package.json-luomiskomento

```
reactApp\client>npm install react -save
```

Kuva 4. Esimerkki paketin asennuskomennosta

Kun tärkeimmät paketit oli asennettu mukaan lukien Webpack, luotiin webpack.config.js tiedosto. Webpack kokoaa tarvittavat moduulit, joita sovellus tarvitsee, ja luo siitä yhden tiedoston näytettäväksi selaimella. Tässä projektissa luotiin bundle.js-tiedosto.

Seuraavassa vaiheessa luotiin index.html. Käyttöliittymä on Reactolle tyypillisesti yhden sivun kokoinen. Tähän html-sivuun luodaan kaikki sovelluksen komponentit. Kuvassa 5 nähdään sovelluksen ainoa html-sivu.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Movie library</title>
    <link rel="stylesheet" type="text/css" href="/src/assets/stylesheets/app.css" />
  </head>

  <body>
    <div class="wrapper"></div>
    <script type="text/javascript" src="bundle.js" charset="utf-8"></script>
  </body>
  <script src="/bundle.js"></script>
</html>
```

Kuva 5. Sovelluksen html-sivu

Seuraavassa vaiheessa luotiin sovelluksen pääkomponenttina toimiva App-komponentti (kuva 6) ja Navigation-komponentti (kuva 7), joka renderöi navigointipalkin (kuva 8). Navigointipalkki liitettiin pääkomponenttiin, ja muut komponentit luodaan lapsikomponenteiksi pääkomponenttiin.

```
import React, { Component } from "react";
import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';
import injectTapEventPlugin from 'react-tap-event-plugin';
injectTapEventPlugin();

import Navigation from './navigation';

export default class App extends Component {

  render() {
    return (
      <MuiThemeProvider>
        <div style={{"width": "100%", "height": "100%"}}>
          <Navigation />
          {this.props.children}
        </div>
      </MuiThemeProvider>
    );
  }
}
```

Kuva 6. App.js

```
import React, { Component } from "react";
import { connect } from 'react-redux';
import { Link } from 'react-router';
import RegisterDialogModal from './register_popup_dialog'

class Navigation extends Component {
  renderLinks() {
    if (this.props.authenticated) {
      return [
        <li key={100}><Link to="logout">Logout</Link></li>,
        <li key={101}><Link to="programmers">Programmers</Link></li>,
        <li key={102}><RegisterDialogModal /></li>,
        <li key={103}><Link to="movies">Movies</Link></li>,
      ];
    }
    else {
      return [
        <li key={1}><Link to="login">Login</Link></li>
      ];
    }
  }

  render() {
    return (
      <div className="frame bit-1 navigation_container">
        <h3 className="bit-40"><img src="./src/assets/images/film_reel.jpg" height="50" width="50" />Movie Library</h3>
        <ul className="bit-60 nav_menu">
          {this.renderLinks()}
        </ul>
      </div>
    );
  }
}

function mapStateProps(state) {
  return {
    authenticated: state.auth.authenticated,
  };
}

export default connect(mapStateProps)(Navigation);
```

Kuva 7. Navigation.js

Kuva 8. Navigointipalkki

Seuraavaksi luotiin routes.js, joka sisältää käyttöliittymän reitityksen eri komponenttien välillä. Tämä komponentti liitettiin html-sivuun kuvan 9 mukaisesti. React Router -paketilla pystytään tekemään reitityksiä näytettäviin komponentteihin (kuva 10).

```
ReactDOM.render(
  <Provider store={store}>
    <Router history={browserHistory} routes={routes} onUpdate={logPageView} />
  </Provider>,
  document.querySelector('.wrapper'));
```

Kuva 9. Routes-komponentin liittäminen html-sivuun

```
import React from 'react';
import { Route, IndexRoute } from 'react-router';

import App from './components/app';
import Programmers from './components/programmers';
import Movies from './components/movies';
import NotFoundPage from './components/notFoundPage';

import Login from './components/auth/login'
import Register from './components/auth/register';
import Logout from './components/auth/logout';
import ForgotPassword from './components/auth/forgot_password';
import ResetPassword from './components/auth/reset_password';

import RequireAuth from './components/auth/require_auth';

export default (
  <Route path='/' component={App}>
    <IndexRoute component={Login} />
    <Route path='login' component={Login} />
    <Route path='programmers' component={RequireAuth(Programmers)} />
    <Route path='movies' component={RequireAuth(Movies)} />
    <Route path='register' component={RequireAuth(Register)} />
    <Route path="logout" component={Logout} />
    <Route path="*" component={NotFoundPage} />
  </Route>
);
```

Kuva 10. Routes.js

Työn muut komponentit luotiin samalla tavalla kuin pääkomponentti ja luotiin reititys komponenteille Routes-komponenttiin (kuva 10). Työssä käytettiin hyödyksi mahdollisimman paljon npm-paketinhallinasta saatavia paketteja, jotta kirjoitettava koodi jäisi mahdollisimman vähäiseksi. Esimerkiksi päänäkymässä (kuva 12) näkyvässä

elokuvalistauksessa käytettiin valmista listakomponenttia (griddle-react), jolla saatiin kirjoitettava koodi muutamaan riviin.

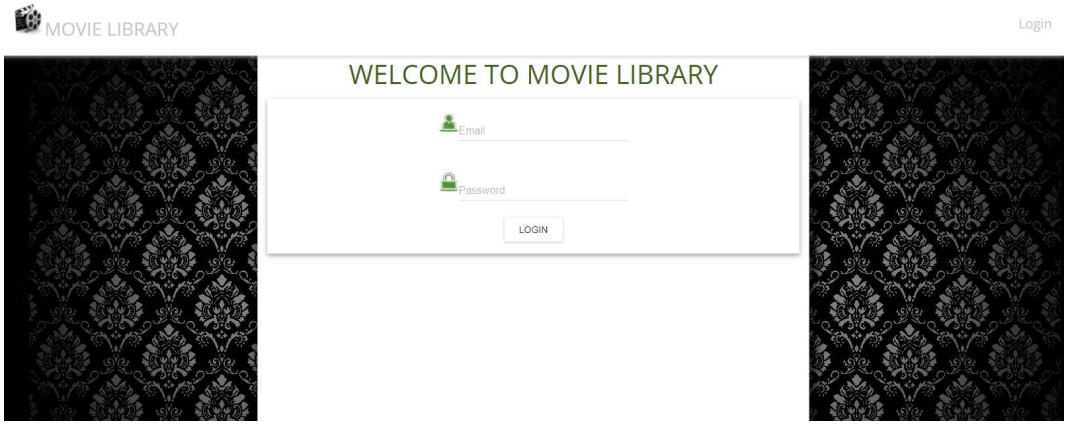
```
<Griddle data={this.props.movieState.movies} plugins={[plugins.LocalPlugin]} sortProperties={sortProperties} components={components}>
    <RowDefinition>
        <ColumnDefinition id="name" title="Name" customComponent={enhancedWithRowData(test)} width={300} />
        <ColumnDefinition id="year" title="Year" width={75} />
        <ColumnDefinition id="imdb" title="IMDb link" customComponent={imdbURL} width={300} />
        <ColumnDefinition id="" title="" customComponent={enhancedWithRowData(deleteButton)} />
    </RowDefinition>
</Griddle>
```

Kuva 11 Esimerkki griddle-react-komponentista

Työ on kokonaisuudessaan katsottavissa ja ladattavissa osoitteissa https://github.com/OKaukola/react-movielibrary-client (client) ja https://github.com/OKaukola/react-movielibrary-server (server).

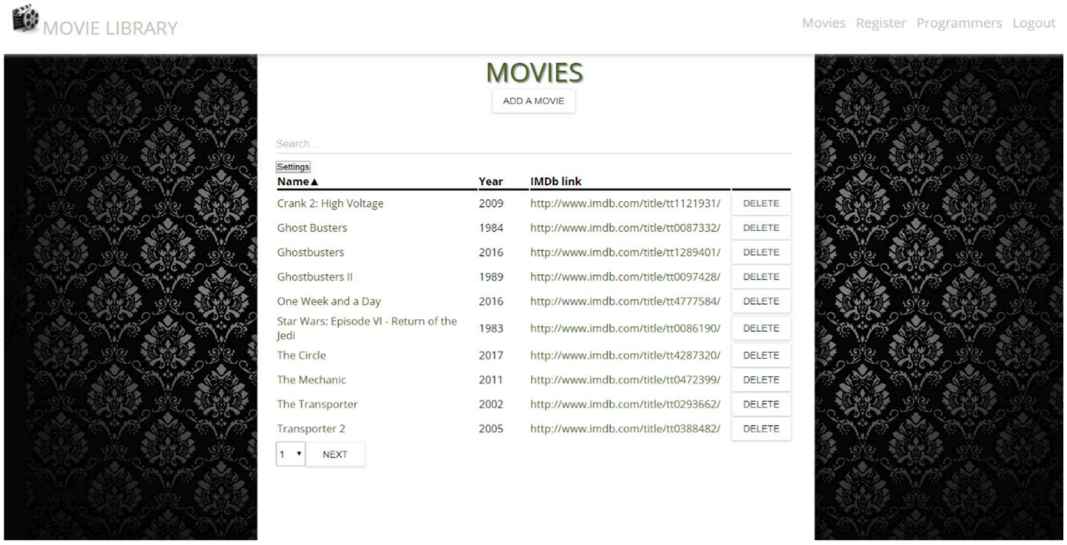
4.2 Visuaalinen näkymä

Käyttöliittymä avautuu sisäänkirjautumissivulle (kuva 12) jossa kysytään käyttäjän käyttäjätunnusta ja salasanaa.



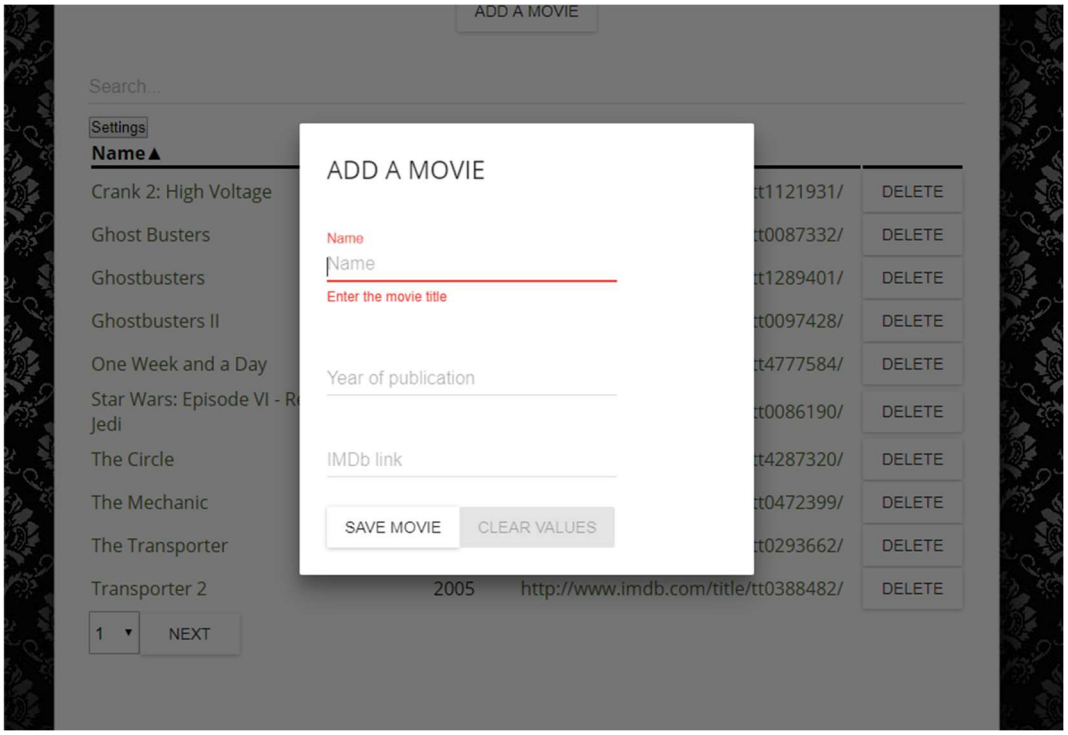
Kuva 12. Käyttöliittymän sisäänkirjautumissivu

Kirjautumisen jälkeen avautuu päänäkymä (kuva 13). Päänäkymässä on painike elokuvien lisäykseen sekä lista elokuvista. Listan asetuksista voi määrittää näytettävien kohteiden määrän sekä näytettävät kentät. Listassa on myös hakuominaisuus sekä jokaisen elokuvan kohdalla painike elokuvan poistamista varten.



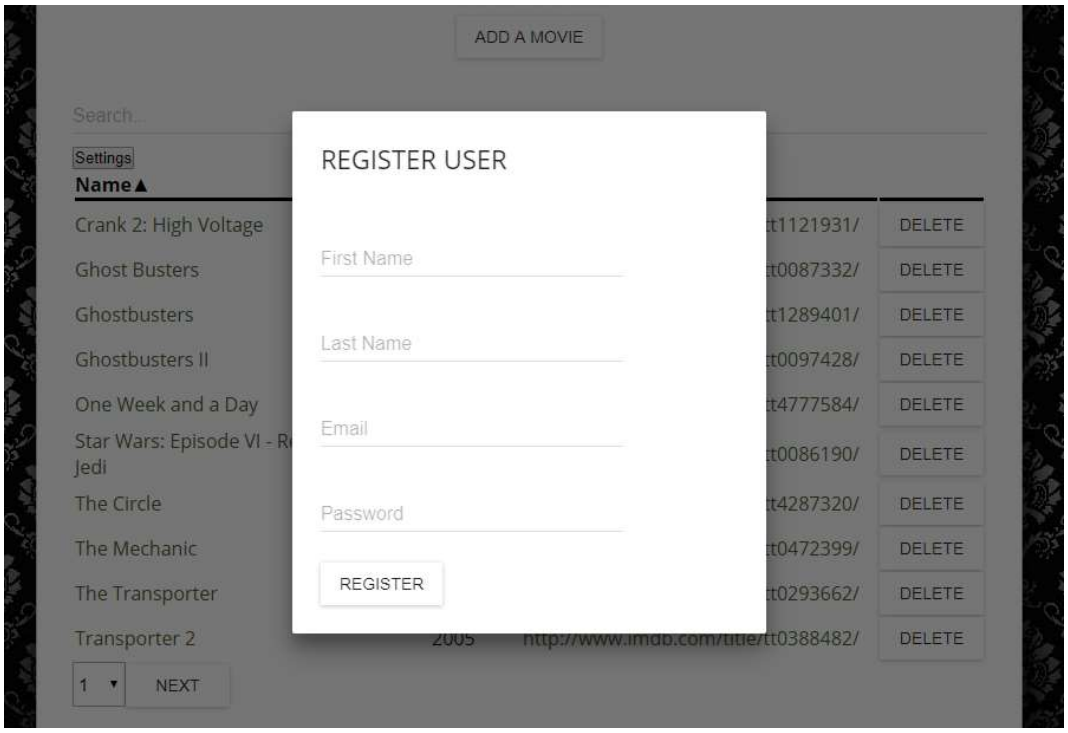
Kuva 13. Käyttöliittymän päänäkymä

Päänäkymässä näkyvästä elokuvien lisäys -painikkeesta avautuu ponnahdusikkuna (kuva 14), johon syötetään elokuvan nimi, julkaisuvuosi ja IMDb (Internet Movie Database) -osoite. Lomakkeessa käytettiin punaista tekstiä antamaan käyttäjälle palautetta vaadituista kentistä. Ikkunassa on kenttien tyhjennykseen ja tallennukseen painikkeet. Ikkunan saa sulkeutumaan napauttamalla ikkunan sivusta.



Kuva 14. Elokuvan lisäys kirjastoon

Uuden käyttäjän lisääminen toimii vastaavasti kuin elokuvan lisäys (kuva 15). Lomake avautuu navigointipalkin Register-kohdasta. Lomakkeeseen annetaan etu- ja sukunimi, sähköpostiosoite ja salasana. Kun käyttäjä on rekisteröity, sähköpostiosoite ja salasana toimivat heti kirjautumisnäkymässä.



Kuva 15. Käyttäjän lisäys rekisteriin

4.3 Jatkokehitysideat

Projektia ohjelmoidessa tuli useita jatkokehitysideoita, kuten elokuvan tietojen haku internetissä toimivasta tietokannasta, istunnon aikakatkaisu, käyttäjäkohtainen tietokanta ja tietokannassa olevien elokuvien muokkaus. Lisäksi mietittiin järjestelmänvalvojalle ainoastaan käyttäjien lisääminen, poisto ja muokkaus, kun nykyisessä versiossa kuka tahansa voi tehdä kyseiset toiminnot. Elokuvien tietojen hakuun voi käyttää esimerkiksi sivustoa https://www.themoviedb.org/, josta löytyy API tietojen hakuun.

5 YHTEENVETO

Insinöörityössä perehdyttiin käyttöliittymä suunnitteluun ja web-käyttöliittymän ohjelmointiin React JavaScript -kirjastoa käyttäen. Työn lukijalle pitäisi jäädä kuva siitä, kuinka React JavaScript -kirjastoa hyödynnetään helposti käyttöliittymää suunnitellessa ja tarpeen tullen sitä osaisi myös soveltaa.

Opinnäytetyössä kehitettiin kirjoittajan omia taitoja käyttää React-kirjastoa, sillä työtä aloittaessa tiedot kirjastosta ja sen käytöstä olivat vähäiset. Tulevaisuudessa kirjoittajan on helpompi lähteä opettelemaan ja kokeilemaan uusia tekniikoita ja ohjelmia.

React-kirjastoa käytettäessä tuli paljon uusia toiminnallisuuksia, joiden opettelu vei runsaasti aikaa. Monipuolisten ja luotettavien lähteiden löytäminen oli ehkä kuitenkin suurin haaste, sillä Reactista ei löytynyt juurikaan tietoa.

LÄHTEET

Bertoli, M. 2017. React Design Patterns and Best Practices. Birmingham UK: Packt Publishing Ltd. Saatavana Packt-palvelusta.

Graafinen. 12.2.2015. Hyvä käyttöliittymä – 10 muistisääntöä. [Verkkosivu]. Graafinen. [Viitattu 24.3.2018.]. Saatavana: http://www.graafinen.com/suunnittelu/digi/hyva-kayttoliittyma-10-muistisaantoa

Hayward, J. 2015. Reactive Programming with JavaScript. Birmingham UK: Packt Publishing Ltd.

International standards for HCI and usability. Ei päiväystä. ISO 9241-11: Guidance on Usability (1998). [Verkkosivu]. UsabilityNet. [Viitattu 25.3.2018]. Saatavana: http://www.usabilitynet.org/tools/r_international.htm

International standards for HCI and usability. Ei päiväystä. ISO 13407: Human-centred design processes for interactive systems (1999). [Verkkosivu]. UsabilityNet. [Viitattu 26.3.2018]. Saatavana: http://www.usabilitynet.org/tools/r_international.htm

ISO 9241-151.2008. Ei päiväystä. ISO 9241-151:2008. [Verkkosivu]. International Organization for Standardization. [Viitattu 26.3.2018]. Saatavana: https://www.iso.org/standard/37031.html

Nieminen, M. Ei päiväystä. Käyttöliittymäsuunnittelu. [Verkkojulkaisu]. Aalto University School of Science. [Viitattu 26.3.2018]. Saatavana: https://mycourses.aalto.fi/pluginfile.php/42664/course/section/17304/CSE-C3800-2015-UI_Design_Guidelines_Marko_Nieminen.pdf

MDN web docs. Ei päiväystä. Introduction to the DOM. [Verkkosivu]. MDN. [Viitattu 26.3.2018]. Saatavana: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Muranen, A. & Harmainen, L. Ei päiväystä. Käyttöliittymä- & käyttäjäkokemussuunnittelu. [Verkkosivu]. IT expertise wiki. [Viitattu 26.3.2018]. Saatavana: https://www.itewiki.fi/opas/kayttoliittymasuunnittelu-ux-user-experience-design-eli-kayttajakokemus/

React. Ei päiväystä. Components and Props. [Verkkosivu]. Facebook Open Source. [Viitattu 3.4.2018]. Saatavana: https://reactjs.org/docs/components-and-props.html

React. Ei päiväystä. Virtual DOM and Internals. [Verkkosivu]. Facebook Open Source. [Viitattu 26.3.2018]. Saatavana: https://reactjs.org/docs/faq-internals.html

React Enlightment. Ei päiväystä. What is React?. [Verkkosivu]. React Enlightment. [Viitattu 26.3.2018]. Saatavana: https://www.reactenlightenment.com/what-is-react.html

ResearchGate. Ei päiväystä. Human centred design process (ISO 13407) with research methods. [Verkkosivu]. ResearchGate. [Viitattu 26.3.2018]. Saatavana: https://www.researchgate.net/figure/Human-centred-design-process-ISO-13407-with-research-methods_fig7_27516496