Mäkiranta Roope

Digitaalisten infonäyttöpäätteiden ohjaamisohjelmiston toteuttaminen hyödyntäen ReactJS-kirjastoa

Opinnäytetyö
Kevät 2019
SeAMK Tekniikka
Tietotekniikan koulutusohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Tietotekniikan koulutusohjelma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Roope Mäkiranta

Työn nimi: Digitaalisten infonäyttöpäätteiden ohjaamisohjelmiston toteuttaminen hyödyntäen ReactJS-kirjastoa

Ohjaaja: Jyri Lehto

Vuosi: 2019 Sivumäärä: 33 Liitteiden lukumäärä: 0

Opinnäytetyön aiheena oleva kehittämistyö suoritettiin tilaajayritykselle Ledimedia Oy:lle. Tutkimuksen tavoitteena oli selvittää parannusmahdollisuuksia jo olemassa olevaan infonäyttöpäätteiden ohjaamiseen tarkoitettuun ohjelmistoon. Pyrkimyksenä työssä oli toimittaa samankaltainen ohjelma, jossa olisi otettu paremmin huomioon käyttäjälähtöisyys ja käyttökokemus. Suunniteltaessa ohjelman rakennetta ja siihen vaadittavia ominaisuuksia otettiin huomioon lainalaisuuksia, joita on todettu hyväksi käyttöliittymäsuunnittelun saralla. Ohjelman toteuttamiseen vaadittava työkalu valittiin oman kokemuksen, sekä muilta käyttäjiltä saadun yleisen positiivisen vaikutelman mukaan. Tutkimuksen loppupuolella vertaillaan alkuperäistä sekä uutta ohjelmaversiota ja pyritään selvittämään, saatiinko vaatimuksien perusteella tuotettua alkuperäistä parempi ohjelma.

Kehitystutkimuksen aiheena olevan ohjelman oli tarkoitus syrjäyttää sen hetkinen Seinäjoen OmaSp Stadionilla oleva infonäyttöpäätteiden ohjaamisohjelmisto.

Avainsanat: kehittämistyö, ohjelmisto, käyttöliittymäsuunnittelu, ReactJS, käyttöliittymä

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Engineering

Author: Roope Mäkiranta

Title of thesis: Producing Control Software for Digital Information Monitors Utilizing the ReactJS-Framework.

Supervisor: Jyri Lehto

Year: 2019 Number of pages: 33

The research in this thesis was made at the request of Ledimedia Oy. The purpose of this thesis was to find improvement possibilities for the control software which was being used with digital information monitors. The goal was to produce equivalent software in which the user and the user experience would have been considered.

In the development process of the software well tried patterns of software development were taken into account. The required tool was chosen based on the general positive impression on it. The original and the new, reproduced improved version were compared in the last part of the thesis and it was considered if the goal was reached with the improvements. The purpose of this thesis was to develop software that could replace the former software used to control the digital information monitors at the OmaSp Stadion in Seinäjoki.

Keywords: Software developing, User interface design, Development study, ReactJS, User interface

SISÄLTÖ

Kuva- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

CSS	Cascading Style Sheets. Työkalu, jolla voidaan tyylitellä HTML-elementtejä.
Framework	Ohjelmoinnissa käytettävä kirjasto, jota hyödyntämällä saadaan ohjelman käyttöön toimintoja ilman, että niitä joudutaan ohjelmoimaan käsin.
Front-End	Käyttöliittymässä käyttäjälle näytettävä näkymä.
Funktio	Tässä yhteydessä ohjelmointikielen toiminnallinen osanen, joka toteuttaa sille annettuja tehtäviä.
Google Chrome	Googlen kehittämä internetselain.
HTML	Hyper Text Markup Language on verkkosivuilla käytettävä merkintäkieli, jolla kuvataan tekstin rakennetta tai esitystapaa.
I/O-operaatio	Input/Output-operaatiot ovat kommunikaatioita laitteista ulos- tai sisäänpäin.
IP	Internetprotokolla, joka määrittää asetukset ja säännöt datapakettien reitittämiseen ja lähettämiseen, jotta datapaketit osaavat oikeaan osoitteeseen.
JavaScript	JavaScript on ohjelmointikieli, jolla toteutetaan komplekseja (monipuolisia) toimintoja verkkosivuilla.
JQuery	Kirjasto, jolla pyritään yksinkertaistamaan ja helpottamaan JavaScriptin toiminnallisuuksien käyttöä.
Komponentti	Tässä yhteydessä komponentilla tarkoitetaan palastellun verkkosivun osaa, joka toimii itsenäisesti. Se voi esimerkiksi sisältää toiminnallisuutta tai tietoa.

LED-screen	Light Emitting Display -screen on näyttöpääte, jossa kuva tuotetaan hyödyntämällä yksittäisiä LED-valoja muodostamaan suuri kokonaiskuva näyttöpinnalle.
Parametri	Parametrilla tarkoitetaan tietoa, joka syötetään ohjelmalle.
Responsiivisuus	Verkkosivupohjaisessa käytössä responsiivisuudella tarkoitetaan verkkosivun mukautuvuutta eri näyttöko´oilla.
SPA	Single-Page Applicationilla tarkoitetaan verkkosivuja, joissa kaikki sisältö latautuu yhdelle samalle verkkosivulle.
SQLite	SQLite on relaatiotietokanta.
UDP	User Datagram Protocol on tiedonsiirtoprotokolla, jolla voidaan siirtää tietoja laitteiden välillä.
XML	Extensible Markup Language. Ladontakieli, jolla säilötään ja kuljetetaan tietoa.

1 JOHDANTO

1.1 Johdantoa

Kehittämistyönä oleva opinnäytetyö tehtiin tilaajayritykselle LediMedia Oy:lle. Pyrkimyksenä oli luoda ohjelma, jolla voitiin ohjata digitaalisia infonäyttöpäätteitä suurina näyttökokonaisuuksina. Opinnäytetyön tekijä on opintosuunnitelmaan kuuluvassa työharjoittelussa tehnyt alkuperäisen jo olemassa olevan ohjelman yhteistyössä ohjaajansa kanssa. Pohdittaessa eri vaihtoehtoja työkaluksi, joilla uuden version voisi rakentaa valikoitui ReactJS-työkalu, joka on JavaScriptin (framework) viitekehykselle rakennettu ohjelmakirjasto.

1.2 Ohjelman toimintaperiaatteesta

Käytössä oleva ohjelma on luotu muun muassa käyttäen web-ohjelmointikieliä kuten JavaScript, HTML ja CSS. Ohjelman pääsivu on HTML-pohjainen web-sivu. Pääsivulla on digitaalisten näyttökokonaisuuksien ohjaukseen painikkeet, jotka lähettävät UDP-komennon mediatoistimelle käyttäen JavaScript-funktiota. Mediatoistimen vastaanotettua UDP-komennon avaimen tietystä IP-osoitteesta, osaa mediatoistin soittaa halutun mainosvideon tai -kuvan näyttöpinnoilla.

1.3 Tutkimuksen tavoitteet

Tutkimuksen tavoitteena oli selvittää mahdollisuuksia jo olemassa olevan ohjelmiston parantamiseen. Parannuksella tavoiteltiin ohjelmiston saamista helppokäyttöiseksi ja selkeäksi, näin ohjelmisto mahdollisesti voisi soveltua myös asiakaskäyttöön sopivaksi tuotteeksi. Olemassa olevan ohjelmiston käyttö vaatii käyttäjältään ohjelmointitaitoja, ja on näin ollen puolivalmiina sopimaton myytäväksi asiakastuotteeksi. Tutkimuksessa pyrittiin selvittämään, miten olisi mahdollista luoda helppokäyttöinen ohjelma, jota osaisi käyttää henkilö, joka omaa perustaidot tietotekniikasta ilman laajaa perehdyttämistä ohjelman käyttöön.

1.4 Ledimedia Oy

LediMedia Oy on vuonna 2012 perustettu osakeyhtiömuotoinen valtakunnallinen yritys, joka on keskittynyt videotekniikkaan. Yritys myy info- ja mainosnäyttöjä, sekä vuokraa näyttölaitteita ja LED-screenejä tapahtumiin. LediMedia Oy tarjoaa palveluita koko näyttöhankinnan elinkaaren ajalle hankinnasta asennuksiin ja huoltoihin. Ledimedia Oy:llä on jo yli kahdenkymmenen vuoden kokemus mainosalalta. Näyttöpintojen hyödyntämisen moniosaamisen lisäksi yritys tarjoaa aineistonhallintaa, pilvipohjaista LEDiMediaCloud-hallintaohjelmistoa, jota voidaan hyödyntää mediaaineistojen esittämiseen erilaisilla digitaalisilla näyttöpinnoilla. (Ledimedia, [viitattu 15.4.2019].)

2 DIGITAALISESTI ETÄOHJATTAVAT INFONÄYTTÖPÄÄTTEET

Digital Signage eli suomeksi käännettynä sisällöltään etäohjattavat digitaalisilla näytöillä (DigitalSignage, 2.9.2018) tarkoitetaan digitaalisia infonäyttöjä, joita voidaan käyttää mainostamiseen tai muun informaation jakamiseen yksityisillä ja julkisilla paikoilla. Informaatiota on mahdollista jakaa hyödyntäen yksittäistä näyttöpintaa tai useista näyttöpinnoista koostuvaa suuren seinäpinta-alan täyttävää kokonaisuutta (Medium, 2.9.2018). Digital Signage -infonäyttöjä käytetäänkin usein mainostamiseen julkisilla paikoilla, joissa ihmiset odottavat jotain, kuten esimerkiksi suojateiden läheisyydessä. Mainostamisen lisäksi infonäytöille suosittuja sijoituspaikkoja ovat ravintolat ja kahvilat, joissa infonäyttöjä on mahdollista käyttää esimerkiksi ravintoloiden ruokalistan näyttämiseen (Medium, 2.9.2018).

2.1 Vaadittava laitteisto

Digitaalisessa mainonnassa käytettävä laitteisto koostuu näyttöpinnasta, mediaprosessorista ja sen ohjaimesta, sekä sisällön toistoon vaadittavasta työasemasta. Laitteiston lisäksi vaaditaan esitettävä sisältö. **Näyttöpintoja** on saatavilla useaa eri kokoa sekä myös teknologioita pinnoille on useita. Näyttöpintojen koko ja käytettävä teknologia valitaan näyttöpinnan sijoituspaikan mukaan. Sijoituspaikan valaistus ja katselualue, jolle materiaalia esitetään, ovat tekijöitä, jotka on otettava huomioon esitettävän sisällön lisäksi koon ja teknologian yhdistelmää valittaessa. **Mediaprosessorin** tehtävä on vastaanottaa esitettävä sisältö työasemalta ja lähettää se muunnettuna oikeaan kokoon näyttöpinnoille (Sinopoli 2010, 97-99.) Sisältöä lähettävänä **työasemana** voidaan käyttää esimerkiksi PC:tä, joka on yhdistetty internetiin, josta saadaan sisältö pilvipalvelun kautta (DigitalSignage, 2.9.2018). Sisällön lähettämiseen voidaan käyttää myös mediatoistinta, jolla voidaan lähettää usealle näyttöpinnalle sama sisältö. Työaseman kautta mainonnan sisältöä voidaan ajastaa ja monitoroida järjestelmää. Näyttöpinnoilla näytettävään **sisältöön** on syytä panostaa, sillä hyvin ajastettu ja suunniteltu mainonta herättää mielenkiinnon katsojassa. Digitaalisessa mainonnassa vaikein osuus onkin oikeanlaisen ja mainontaa nauttivan henkilön huomiota ylläpitävän sisällön tuottaminen. (Sinopoli 2010, 99-100.)

3 KÄYTTÖLIITTYMÄN SUUNNITTELU

Ennen varsinaisen käyttöliittymän tekemistä, on hyvä selvittää edellisen käyttöliittymän korjaustarpeet. On myös mahdollista tutkia kilpailijoiden palveluita, ja etsiä niistä korjaustarpeita, jotta osattaisiin välttää myös heidän virheidensä toistaminen. (Sinkkonen, Nuutila & Törmä, 2009, 285.)

Tuotesuunnittelun aikana valmistuneesta kehitelmästä pyritään poistamaan asiat, jotka eivät ole kunnossa. Kehitelmän perusteella aloitetaan käyttöliittymän teko. Ennen kuin tuotetta julkaistaan, on selvitettävä, täyttyykö tuotteelle annetut vaatimukset käytettävyydessä. (Sinkkonen, Nuutila & Törmä 2009, 285.)

3.1 Käytettävyys ja käyttökokemus

Käsite käytettävyys kuvaa palveluissa käyttäjän saamaa kokemusta palvelun käytöstä. Käytettävyys kuvaa palvelussa käyttölaatua, joka on joko positiivinen tai negatiivinen. Palvelu voi esimerkiksi olla hankalasti ymmärrettävissä tai visuaalisesti miellyttävä. Ne vaikuttavat käyttäjän käyttökokemukseen. Hyvän käytettävyyden perustana vaaditaan käytettävyyden olevan sillä tasolla, että ohjelmaa on miellyttävä käyttää ja käyttäjän on oltava tyytyväinen palvelun käyttöön. Käytettävyys ja käyttökokemus ovat toisistaan riippuvaisia, ne ovat näin kaksisuuntaisesti sidoksissa toisiinsa. Mikäli käyttäjä pitää palvelun ulkonäöstä ja kokee sen miellyttäväksi, on mahdollista, että käyttäjä antaa anteeksi muutamat käytettävyyden virheet. Sama pätee myös toisinpäin, palvelun ollessa erittäin kätevä, ei sen ulkonäköön kiinnitetä niin paljon huomiota. Käyttötilanteessa on anteeksi annolla myös rajansa. Vaikka kuinka visuaalisesti miellyttävä ja ulkoisesti kaunis palvelu olisikin, mikäli se ei toimi kriittisellä hetkellä, käyttäjä saa huonon kokemuksen palvelun käytöstä ja tilanne voi muuttua tunnepitoiseksi. (Sinkkonen, Nuutila & Törmä 2009, 18-19.)

Käyttöliittymää suunniteltaessa on otettava huomioon ihmisen fyysiset piirteet ja kognitiiviset ominaisuudet. Suuret kontrastit keskittävät ihmisen huomion. Asioiden, kuten esimerkiksi painikkeiden, on oltava loogisessa järjestyksessä. Mikäli toiminnallisuus ei sovi kyseiseen tehtävään, ja esimerkiksi tietosisältö ei koske tehtävää, saa käyttäjä palvelusta negatiivisen käyttökokemuksen. (Sinkkonen, Nuutila &

Törmä 2009, 18-19.) Käyttöliittymää suunniteltaessa onkin äärimmäisen tärkeää muistaa, missä tehtävässä ja tilanteessa palvelua käytetään, sekä missä ympäristössä ja millaiselle käyttäjälle palvelu on suunnattu (Sinkkonen, Nuutila & Törmä 2009, 21).

Käytettävyyttä voidaan myös ymmärtää terminä käyttökelpoisuus. Käyttökelpoisuudella tarkoitetaan palvelun helppokäyttöisyyttä ja käytön tehokkuutta. Hyvin käytettävän palvelun käyttöön on helppo päästä sisälle ja käyttää palvelua tehokkaasti, jo heti ensimmäisellä kerralla. Palvelun tehokkuutta heikentävät ihmisen tekemät virheet. Virheen teko joko huomataan lopussa, jolloin joudutaan palaamaan askelia takaisin, tai heti, jolloin virhe korjataan välittömästi. Virheet lopputulemassa rikkovat vaatimusta virheettömyydestä. Heti korjatut virheet heikentävät palvelulle annettua vaatimusta käytön tehokkuudesta. (Sinkkonen, Nuutila & Törmä 2009, 18-19.)

3.2 Käyttötarinat järjestelmän korjaamisessa

Suunniteltaessa uutta käyttöliittymää käyttötarinat ovat hyvä tapa jäsentää käyttäjän tapahtumakulkua palvelua käyttäessä. Vanhaa parannettaessa tai korjattaessa on myös hyvä tehdä käyttötarinat tulevasta uudesta sekä vanhasta jo käytössä olevasta palvelusta. Kuvauksessa, miten ohjelma toimii, aloitetaan asettamalla **tavoitteet** ja määrittämällä mihin yritetään päästä uutta palvelua tehdessä tai vanhaa parannettaessa. **Korjauksen laadun** määrityksessä päätetään, korjataanko vanhaa esimerkiksi lisäämällä toimintoja vai vaaditaanko palvelussa olevan ongelman korjaamiseen aivan uuden ohjelman tekoa. **Persoonalla ja tavoitteella** pyritään personoidun tarinan kautta tuomaan esille, miksi on kohdattu tarve parannuksille. **Toimintatarinasta** tulee ilmi millaisia vaiheita nykyisen palvelun käyttäjä kohtaa. Toimintatarinalla esitellään myös mahdollisia käyttäjän kohtaamia ongelmia palvelun käytössä. **Käyttötarinassa** taasen esitetään ehdotus, millaiseksi ohjelma muutetaan, jotta palvelun käyttö olisi jouhevaa, eikä käyttäjän tarvitsisi kohdata samoja ongelmia kuin aiemmassa palvelussa. (Sinkkonen, Nuutila & Törmä 2009, 175-177.) Taulukossa 1 havainnollistetaan käyttötarinoiden käyttöä esimerkin avulla. Esimerkkinä toimii opinnäytetyössä aiheena oleva ohjelma, jota pyritään parantamaan helppokäyttöisemmäksi.

Taulukko 1. Käyttötarinan kuvaus

Yritys ja sen tavoite	Pyritään saamaan käyttöön helppokäyttöinen ohjelma, jota olisi mahdollista osata käyttää ilman laajaa koulutusta.
Korjauksen laatu	Rakennetaan vanhan perusteella uusi ohjelma.
Persoona ja tavoite	Sami tarvitsisi käyttöönsä digitaalisten infonäyttöjen ohjausohjelman, jolla voisi ohjata kohdennettuja mainoksia urheilustadionin laidoilla olevilla näyttöpinnoilla. Kuitenkin tämän hetkisen ohjelman käyttö vaatii liikaa Samin tietotekniseen taitotasoon nähden, eikä hänellä ole aikaa perehtyä syvemmin asiaan.
Toiminta nyt eli toimintatarina	Samin on osattava verkkosivu-ohjelmointia ja HTML-kieltä, jotta pystyisi päivittämään käyttöliittymässä olevia painikkeita, joilla mainosten ohjaus tapahtuu. Sami joutuu ohjelmoimaan joka kerta uudestaan painikkeita saadakseen painikkeille haluamansa tekstit. Mikäli mainoksia on enemmän kuin ohjelmassa sillä hetkellä, on Samin ohjelmoitava uusi painike käyttöliittymään. Samin luomalle uudelle painikkeelle on ohjelmoitava toiminnallisuus.
Toiminta uudella järjestelmällä eli käyttötarina	Sami avaa ohjelman. Käyttöliittymässä on painike, jolla luodaan uusia painikkeita tarvittavissa oleva määrä. Sami painaa painikkeesta, jolloin aukeaa lomake, jossa kysytään kysymyksiä toiminnallisuutta koskevista kysymyksistä. Painettaessa OK, ohjelma luo uuden painikkeen käyttöliittymään, joka sisältää toiminnallisuutta perustuen siihen, mitä lomakkeesta on annettu vaatimuksia.

3.3 Käytettävyyden arviointi

Ohjelman käytettävyyden arvioimiseen on Jakob Nielsen muodostanut 10 kohdan listan, joiden noudattamista pidetään yleisesti hyvänä tapana suunniteltaessa ohjelmaa. Arvioinnin kriteerejä kutsutaan heuristisiksi, sillä ne ovat ns. nyrkkisääntöjä, eivätkä ennalta määrättyjä standardeja. (Nielsen Norman Group, 24.4.1994.)

Käyttäjän hämmennyksen välttämiseksi ohjelmaa käytettäessä on tärkeää, että käyttäjälle tarjotaan tarpeeksi informaatiota siitä, mitä ohjelmassa on tapahtumassa juuri sillä hetkellä. Sivua ladattaessa näytetään kuvake sen merkiksi tai mikäli verkkokaupassa on vaatekoko loppu, käyttäjälle tuodaan asia ilmi. Tätä kutsutaan **läpinäkyvyydeksi ohjelman tilassa.** (Nielsen Norman Group, 24.4.1994.)

Ohjelman käyttömukavuuden takaamiseksi on ohjelmassa käytetyn kielen oltava käyttäjäläheistä. Käytetyn sanaston ja lauseiden on oltava yksiselitteisiä ja tuttuja, ja on pyrittävä välttämään teknistä sanastoa. **Ohjelman ja tosimaailman** välille on luotava yhteys, joka tekee ohjelman käytöstä luonnollista ja loogista. (Nielsen Norman Group, 24.4.1994.)

Käyttäjillä on ohjelmia käyttäessään taipumus tehdä virheitä. Virheiden tapahtuessa on käyttäjälle hyödyllistä tarjota informaatiota virheestä, jonka perusteella käyttäjä **huomioi** tapahtuneen virheen, pystyy **määrittelemään** ja ymmärtämään, missä ja miksi virhe on tapahtunut, sekä osaa **selviytyä** virhetilanteesta. Tarjottu informaatio on tuotava käyttäjälle selkokielisenä tekstinä, jossa osoitetaan selvästi esiintyneen virheen syy ja rakentavasti esitetään ratkaisuehdotus ongelmalle. Käyttäjälle onkin tarpeellista antaa **vapaus ja kontrolli** virhetilanteen sattuessa. Ohjelmaan tehdään itsestään selvä ”hätäpoistumistie”, kuten painike, jota pitkin on virhetilanteiden sattuessa mahdollista palata askelia takaisin ja peruuttaa tehty virhe. Tapahtuvien **virheiden ennaltaehkäisy** on erinomainen tapa parantaa ohjelman käytön tehokkuutta. Virheitä voidaan ehkäistä näyttämällä virheilmoituksia, mutta vielä parempi tapa on hoitaa ohjelman suunnittelu niin, ettei virheisiin ajauduta lainkaan. Virheiden tapahtuminen voidaan estää esimerkiksi käymällä dialogia käyttäjän kanssa; ”Oletko varma, että haluat jatkaa?”, johon käyttäjä vastaa kieltävästi tai myöntävästi. (Nielsen Norman Group, 24.4.1994.)

Ohjelmalta vaaditaan **johdon- ja standardinmukaisuutta.** Käyttäjälle ei tulisi tulla vastaan tilanteita, jotka aiheuttavat pohtimista. Eri sanojen, tilanteiden tai toimien tulee johtaa samaan lopputulokseen (Nielsen Norman Group, 24.4.1994). Käytetyn terminologian vaikutusta selventää tutkimus, jossa talviurheiluun keskittyvän verkkokaupan perinteiseen termistöön kuuluva ostoskori vaihdettiin ostoskelkaksi. Uusi nimipolitiikka aiheutti hämmennystä asiakkaissa ja siitä luovuttiin (Nielsen Norman Group, 22.8.1999).

Helppokäyttöisyyteen ja tehokkuuteen pyrittäessä ohjelman käytön on oltava sulavaa. Käyttäjän on tiedostettava heti, mikä painike johtaa mihinkin lopputulokseen. Käyttäjillä on usein jotain aiempaa kokemusta käyttöliittymistä ja niistä opittua käsitystä toiminnallisuudesta. Nämä aiemmat kokemukset luovat ennakkokäsitystä myös seuraavan ohjelman käyttöön. Tämän vuoksi ohjelmaa suunniteltaessa on tarpeellista huomioida ja verrata **mieleen palauttamisen** ja **tunnistamisen eroja.** Ohjelman käyttö on tehokasta jo ensimmäisillä kerroilla, mikäli käyttäjä tunnistaa samankaltaisia toimintatapoja ja käyttömenetelmiä kuin aiemmin käytetyissä ohjelmissa. (Nielsen Norman Group, 24.4.1994.)

Ohjelman käyttäjiin kuuluu usein eri tietoteknisellä tasolla olevia käyttäjiä. Jotta ohjelman käyttö olisi miellyttävää monen tasoisille osaajille, olisi ohjelmaan sopivaa lisätä ominaisuuksia, jotka nopeuttavat ohjelman käyttöä kokeneelle käyttäjälle. Ohjelman suunnitteleminen **joustavaksi** ja käytön **hyötysuhteen** korkealle saamiseksi voidaan esimerkiksi käyttäjälle antaa mahdollisuus modifioida useimmiten käytettyjä ominaisuuksia. (Nielsen Norman Group, 24.4.1994.)

Esteettisillä ja **minimalistisilla** valinnoilla käyttöliittymäsuunnittelussa saadaan käyttäjän huomio pidettyä relevanteissa tehtävän suoritukseen liittyvissä seikoissa. Ohjelman dialogien täyttäminen irrelevantilla informaatiolla tai muotoiluvalinnoilla vie ohjelman käytöstä tehokkuutta, sillä muutoin relevantti informaatio joutuu taistelemaan huomiosta irrelevantin informaation kanssa. (Nielsen Norman Group, 24.4.1994.)

Ohjeistus ja **dokumentaatio**. Siitä huolimatta, että yleisesti pidetään parempana, että järjestelmää olisi mahdollista käyttää ilman dokumentaatiota, voi olla kuitenkin tarpeellista tarjota apua käyttöön ohjeistuksilla ja dokumentaatioilla. Käytön avuksi

laaditut dokumentaatiot tulee luoda helppolukuisiksi ja käyttäjäkeskeisiksi. Niissä tarjotaan apua käyttäjän pulmiin askel askeleelta. Dokumentaation pituuden on pysyttävä maltillisena. (Nielsen Norman Group, 24.4.1994.)

4 REACTJS

ReactJS on Facebookin luoma JavaScript-ohjelmointikielen kirjasto, joka on kehitetty verkkosivupohjaisten käyttöliittymien tekoon (w3schools, [viitattu 11.4.2019]). Kirjasto on julkaistu yleisölle käyttöön 2013 (Github, [viitattu 15.4.2019]). ReactJS-kirjaston suosio on muodostunut sen ominaisuudelle luoda interaktiivisia käyttöliittymiä kivuttomasti ja vähemmällä vaivalla kuin edellisillä käytössä olleilla työkaluilla. Kirjastoa käytettäessä verkkosivu muodostuu paloista ja jokaista palaa kohden on oma komponenttinsa, jota käsitellään vain, kun kyseiselle komponentille on tarvetta, esimerkiksi datan päivittyessä kentässä tai listaa täydennettäessä. (React, [viitattu 13.4.2019]). Alla olevassa kuvassa on pyritty havainnoimaan, kuinka ReactJS-kirjastossa hyödynnetään komponenttien palastelua verkkosivulla. Jokainen verkkosivulla oleva pala on komponentti, joka toimii omana autonomisena itsenään (React, [viitattu 13.4.2019]). ReactJS-kirjaston suosiosta kertoo sitä käyttävät verkkosivut ja sovellukset. Sen käyttöä hyödyntää muun muassa Facebook, Instagram ja Airbnb (Medium, 16.4.2019).

Kuva 1. ReactJS-verkkosivun esimerkkipohja

4.1 Multi-Page Design vs Single-Page App Model

Aikaisemmin verkkosivupohjaisessa ohjelmoinnissa käytettiin tapaa, jossa jokainen verkkosivulla oleva alisivu rakennettiin omaksi itsenäiseksi verkkosivukseen. Monen itsenäisen sivun tapaa kutsutaan yleisesti termillä **Multi-Page Design.** ReactJS-kirjaston renderöidessä yksittäisen muokkauksen alla olevan komponentin vanhentuneella usean sivun verkkosivutyylillä jonkin muuttuessa, joudutaan koko sivu lataamaan aina uudestaan, tämä aiheuttaa ylimääräistä odottelua, joka huonontaa käyttäjän käyttökokemusta. (Chinnathambi, 2017, 2-3.)

Modernia uuden tyylin yksittäisen sivun mallia kutsutaan nimellä **Single-Page App** (SPA). SPA on malli, jossa pysytään koko sivulla oloaika yhdellä yksittäisellä sivulla. SPA eroaa Multi-Page Designista juuri sillä, että poistumiseen muille sivuille tai uudelleen sivun lataamiselle ei ole tarvetta. Sen sijaan ne sivun osat, jotka vaativat uudelleen latausta esimerkiksi tiedon muututtua, latautuvat ainoastaan. (Chinnathambi, 2017, 3-4.)

4.2 JSX

JSX eli JavaScript XML on ReactJS-kirjaston oma muotorakenne. JSX yhdistää JavaScriptiä ja HTML-merkintäkieltä. XML-muotorakenteen yhdistävä JSX esikäsittelee ohjelmoidun merkkikoodin ymmärrettävään muotoon JavaScriptille. (Christopher, 2016, 6-7). JSX-rakenteen käyttö ei ole pakollista ReactJS-kirjaston yhteydessä, mutta on erittäin suositeltavaa, sillä käytöllä saadaan ohjelmakoodista huomattavasti hienostuneempaa ja mutkattomampaa (React, [viitattu 18.4.2019]).

4.3 Komponentit

ReactJS-kirjaston **komponentit** tarjoavat mahdollisuuden jakaa käyttöliittymä yksittäisiin paloihin, jotka toimivat autonomisesti. Palat ovat itsenäisiä ja uudelleen käy-

tettäviä komponentteja, jotka toimivat eristyksissä toisistaan, ellei niiden välille muodosteta yhteyksiä. Komponentit ReactJS-kirjastossa toimivat samaan tapaan kuin normaalit JavaScript-funktiot. Niille voidaan antaa ominaisuuksia ja arvoja, ja ne palauttavat halutun tiedon käyttöliittymää käyttävälle päätteelle. (React, [viitattu 15.4.2019].) Komponenteilla on ominaisuus, jota kutsutaan elinkaareksi. Sillä kuvataan komponenttien elämää, joka alkaa komponentin synnystä, sen päivittämisestä ja lopulta, kun komponentti on tehnyt tehtävänsä ja sitä ei enää tarvita, se ”kuolee”. Komponentin elinkaarella pyritään käyttämään annetut resurssit mahdollisimman tehokkaasti. (Bits and Pieces, 22.8.2018.) Esimerkiksi vaihdettaessa toisesta applikaatioista toiseen, on huomattavan epäsuotavaa jättää edellisenä käytetty applikaatio taustalle kuluttamaan resursseja kuten prosessoria tai akkua, mikäli sen tarjoamia palveluita ei tarvita enää. Sama pätee komponentteihinkin, jotka herätetään henkiin, pelkästään kun niille on tarve. Komponenteille tyypillisiä käynnistymisaikoja ovat esimerkiksi sivua avattaessa tai sivulla olevia lomakkeita täydennettäessä. Komponentti pysyy hereillä täydennyksen ajan ja sivu kuolee suljettaessa. (React, [viitattu 12.4.2019].)

4.3.1 Props - ominaisuudet

Arvoja ja ominaisuuksia kutsutaan **propseiksi**, joka on lyhenne sanasta properties - ominaisuus. ReactJS-kirjastossa propseja hyödynnetään lähettämään dataa komponenteille. Ne toimivat samaan tapaan kuin parametrit normaaleissa JavaScript-funktioissa. Parametrit ja propsit ovat muuttumattomia, sillä ne ovat kehitetty samaan tyyliin kuin JavaScript funktiot. (Agiliq, 25.5.2018.)

4.3.2 State - tila

ReactJS-komponenteilla on ominaisuus **state**, joka pitää sisällään tiedon siitä, missä tilassa komponentti on. Statet ovat komponenttien datavarastoja. Niitä käytetään komponenttipalojen päivittämiseen käyttäjän käyttäessä toimintoja kuten painiketta painettaessa. ReactJS-kirjastossa on kahden tyyppisiä komponentteja: tilal-

lisia ja tilattomia. Funktioihin perustuvat normaaliin JavaScriptiin verrattavat komponentit ovat tilattomia. Tilallisiksi luetaan luokkiin perustuvat ja periytyvät komponentit. (Agiliq, 25.5.2018.)

4.4 NodeJS-palvelin

ReactJS-sovelluksia taustalla pyörittävänä moottorina on tavallista käyttää NodeJS-palvelinta. Se on JavaScript-pohjaisten sovellusten ajoon suunniteltu avoimen lähdekoodin ympäristö, joka ylittää alustarajoitukset, ja on näin ollen on käyttöjärjestelmäriippumaton. Ajettaessa NodeJS-palvelimella JavaScriptiin perustuvia ohjelmia, saadaan ohjelmasta erittäin suorituskykyinen. Sovellus, jota ajetaan käyttäen NodeJS-palvelinta, on yksittäinen prosessi, eikä esimerkiksi uusien kyselyjen tullessa muodosta uutta säiettä jokaiselle erilliselle kyselylle. Säikeiden muodostamatta jättämisellä saadaan vältettyä virheiden esiintymistä useiden kyselyjen esiintyessä yhdenaikaisesti. Dataa syötettäessä tai tulostettaessa NodeJS palaa prosessoimaan tarvittavaa dataa vasta, kun se saa vastauksen käytön kohteena olevalta palvelulta. Säikeitä tai prosessointisyklejä käytettäessä ohjelma jäisi odottamaan datan prosessoimista ja tukkisi muun liikenteen sen ajaksi. Tämä toimintatapa antaa mahdollisuuden käsitellä tuhansia samanaikaisia yhteyksiä hyödyntäen yksittäistä palvelinta ilman tarvetta huolehtia kuormasta säikeillä. (Nodejs, [viitattu 15.4.2019].) Tavanomaisesti NodeJS-palvelinta ohjataan käyttäen joko tekstieditorin tai tietokoneen komentoriviä, jolla käynnistetään se isännöimään sovellusta (Nodejs, [viitattu 16.4.2019]).

5 DIGITAALISTEN INFONÄYTTÖJEN OHJAUSOHJELMA

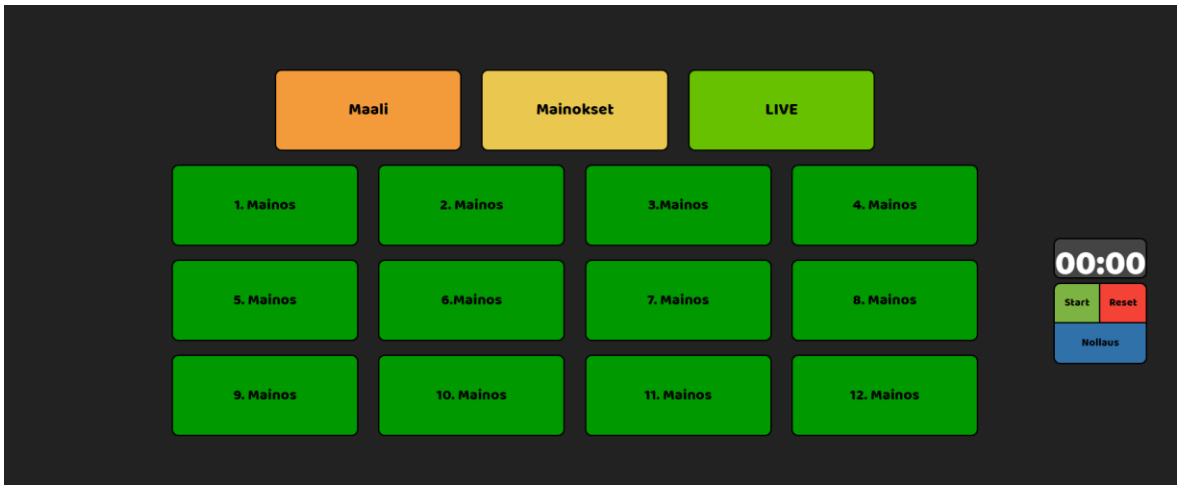
Työn tavoitteena oli toteuttaa ohjelma digitaalisten infonäyttöpäätteiden ohjaamiseen. Ohjelma toteutettiin verkkosivupohjaiseksi hyödyntäen JavaScriptin ReactJS-kirjastoa. Ohjelman käyttöliittymää suunniteltaessa tavoitteena oli luoda mahdollisimman selkeä ja helppokäyttöinen käyttöliittymä.

5.1 Vaatimusmäärittely

Ennakkovaatimuksina ohjelmalle oli tehdä vanhaa huomattavasti helppokäyttöisempi ohjausohjelma, jota pystyisi käyttämään ilman suurempaa tietoteknistä osaamista ja jonka käyttöön ei välttämättä tarvittaisi erillistä käyttökoulutusta normaalin perehdyttämisen lisäksi. Ohjelmasta olisi tultava selkeästi esille sen perusominaisuudet ja niiden käyttötarkoitus. Ohjelman perusominaisuuksilla tarkoitetaan mahdollisuutta lisätä käyttötarpeen mukaan painikkeita käyttäen tarvittavia parametrejä sekä poistaa niitä.

5.2 Edellinen versio

Digitaalisten näyttöpäätteiden ohjaamiseen käytetty ensimmäinen versio oli ulkonäöltään selkeä ja sisälsi ominaisuuden esittää mainoksia infonäyttöpäätteillä painikkeita painamalla sekä mahdollisuuden ajanottoon sekuntikellolla. Tarve ohjelman uudistamiseen tuli sen vaikeakäyttöisestä painikkeiden ominaisuuksien muuttamisesta mainoksien vaihtuessa eri tapahtumiin. Jokainen painike on erikseen ohjelmoitu ohjelmakoodiin ns. ”kovakoodattuna”, mikä toi ongelmia ohjelmointia osaamattomalle käyttäjälle. Ohjelman sisältämien painikkeiden muuttamiseen vaadittiin osaamista muun muassa HTML-merkkikielen osaamisesta ja JavaScriptistä.



Kuva 2. Alkuperäinen käyttöliittymä

5.3 Työn toteutus

Opinnäytetyön projekti alkoi kartoittamalla kehitystarpeita ja päättämällä eri ominaisuuksien tarpeesta uudessa versiossa yhdessä tilaajayrityksen kanssa. Kehitystarpeiksi valittiin helppokäyttöisyyteen painottaminen ohjelmaa luotaessa, sillä edellinen käytössä oleva versio koettiin liian monimutkaiseksi. Ohjelmaa tulisi pystyä muokkaamaan tilanteen ja tapahtuman vaatimalla tavalla.

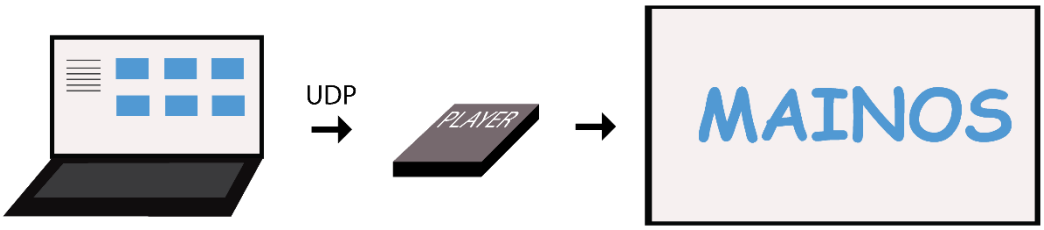
Projektin ohjelman toteuttamiseen valikoitui työkalu, josta opinnäytetyön tekijällä oli jo hieman aiempaa kokemusta. Työkalulla oli myös yleisesti positiivinen maine hyvänä käyttötarpeeseen sopivana ratkaisuna.

Toteutukseen sopivan työkalun ja kartoitettujen kehitystarpeiden myötä lähdettiin suunnittelemaan käyttöliittymää ohjelmalle. Ymmärrystä ja tuntemusta hyvän käyttöliittymän toteutukseen saatiin kirjallisuudesta. Suuri osa ajasta meni ReactJS-työkalun ominaisuuksien ja käytön opiskeluun.

5.4 Ohjelman toimintaperiaate ja laitteisto

Mainoksiin kuuluvat materiaalit on ennalta asetettu mediatoistimelle. Ohjelman kautta mainoksia ohjattaessa käyttäjä valitsee soitettavan mainoksen käyttöliittymästä painiketta painamalla. Painikkeen painallus pyytää ohjelmaa lähettämään

UDP-yhteydellä painikkeelle asetettujen parametrien mukaiset tiedot mediatoistimelle, joka vastaanottaa tiedot ja käsittelee niistä tarvitsemiansa. Jokaisen painikkeen parametreihin kuuluu kolme tietoa: Nimi, Key ja IP. Parametriä Nimi käytetään painikkeen nimeämiseen. Key eli avain on yhteys painikkeen ja mediatoistimella sijaitsevan mainostiedoston välillä. Mainonnan materiaaleja lisättäessä jokaiselle materiaalille annetaan personoiva avain, jonka perusteella on mahdollista kutsua ulkoisella ohjelmalla materiaalia soitettavaksi näyttöpäätteellä. IP sisältää mediatoistimelle asetetun IP-osoitteen, johon UDP-yhteyttä hyödyntäen lähetetään parametrit ohjelmasta. Ohjelman ohjaukseen käytettävä tietokone ja mediatoistin on asetettu samaan paikalliseen verkkoon. Alla olevassa kuvassa on pyritty havainnollistamaan komentoketjua, joka vaaditaan halutun mainoksen valitsemisesta materiaalin näyttämiseen näyttöpäätteellä.



Kuva 3. Vaadittavat toimet

5.4.1 Ohjelman tyylittely

Ohjelman tyylittelyssä käytettiin Bootstrapia, joka on ilmainen front-end-kirjasto, jolla pyritään helpottamaan ja nopeuttamaan verkkosivujen toteutusta. Kirjasto tarjoaa HTML-kieleen ja CSS-tyyleihin perustuvia tyylipohjia. Tässä työssä kirjastoa on hyödynnetty käyttämällä tyylittelyjä painikkeille sekä lomakekentille. Kirjasto tarjoaa myös tyylittelyjä moneen muuhun tarpeeseen (w3schools.com, [viitattu 15.4.2019]).

5.4.2 Painikkeiden tallennus

Painikkeiden tallentamiseen myöhempää käyttöä varten vaadittiin ohjelmaan tietokanta, joka säilöisi tiedot painikkeista. Ilman tietokantaan tallennusta painikkeet katoaisivat ohjelmasta sivua päivitettäessä. Tietokantayhteys SQLiteen muodostetaan käyttäen NodeJS-palvelinta, jota käytetään myös ReactJS-kirjastossa. Käytetyksi tietokannaksi valikoitui SQLite. Se on kevyt, omaa laajan käyttäjäpohjan sekä on luotettava vuoksi (w3schools.com, [viitattu 15.4.2019]).

5.4.3 UDP-komennon lähetys

UDP-komennolla lähetetään ohjelman painikkeilta saadut tiedot mediatoistimelle, joka käsittelee, tietoja ja soittaa niiden perusteella oikean mainosmateriaalin näyttöpäätteillä. UDP-komento hyödyntää lähetyksessä ohjelmassa annettua IP-osoitetta, jonka mukaan painikkeen tiedot osataan ohjata oikeaan osoitteeseen mediatoistimelle. UDP-komennon lähetykseen käytetään jQueryä. Se on ominaisuuksiltaan rikas, mutta pienikokoinen ja nopea JavaScript-kirjasto (jQuery, [viitattu 15.4.2019]).

5.5 Lopputulos

Työn lopputuloksena saatiin tuotettua helppokäyttöinen ja selkeä ohjelma, jossa on mahdollista luoda ja poistaa painikkeita, sekä käyttää ohjelmaa digitaalisten infonäyttöpäätteiden ohjaamiseen. Painikkeiden tiedot tallennetaan paikalliseen tietokantaan, josta aikaisemmin muodostettujen painikkeiden tiedot haetaan näkymään verkkosivua avattaessa. Ohjelma pyrittiin toteuttamaan mahdollisimman selkeäksi, ja se sisältääkin kuvaavaa otsikointia sekä vihjeitä, jotka auttavat käyttäjää suoriutumaan painikkeiden luonnista ohjelmaan. Viemällä kursori yksittäisten lomakekenttien päälle aukeaa vihjelaatikko, joka sisältää esimerkin tai lausemuotoisen vihjeen siitä, mitä lomakekentän kuuluisi sisältää. Ohjelman värimaailmassa on huomioitu yleiset ennakkokäsitykset värien suhteen. Painikkeiden lisäys tapahtuu painamalla vihreää painiketta. Vastaavasti poistopainike on toteutettu keltaisella värillä korre-

loiden ylimääräisen huomioinnin tarvetta. Ohjelmallisesti luodut painikkeet muodostuvat ruudukkoon otsakkeen ”Luodut painikkeet:” alle, josta ne ovat valmiita käyttöön.

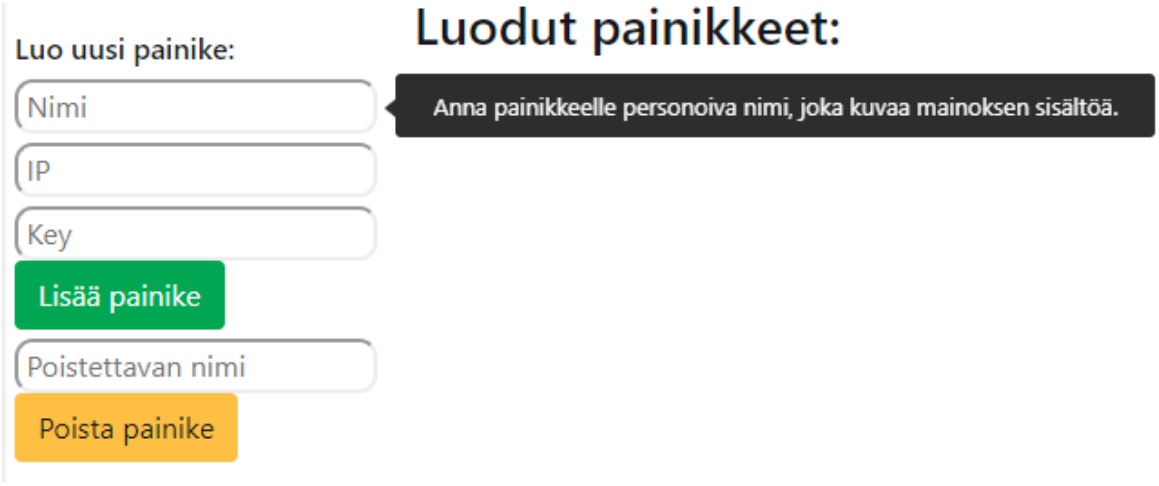
Kuva 4. Digitaalisten infonäyttöpäätteiden ohjaamisohjelmisto

5.6 Ohjelman käyttö

Ohjelman sisältämistä ominaisuuksista tärkeimmät ovat painikkeiden lisäys sekä poisto. Näiden käyttöä demonstroidaan vaihe vaiheelta seuraavissa luvuissa.

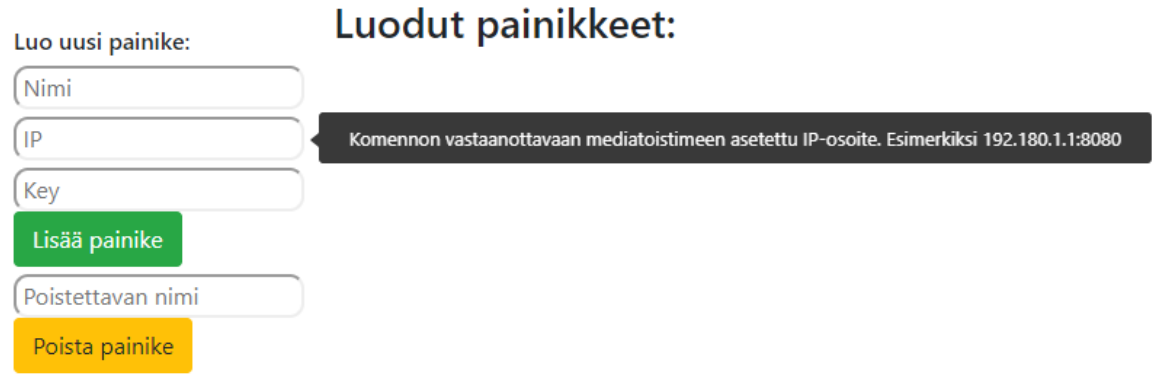
5.6.1 Painikkeen lisääminen ohjelmaan

Painikkeen lisäämien aloitetaan antamalla personoiva nimi painikkeelle. Viemällä kursori Nimi-lomakekentän päälle aukeaa vihje vaatimuksista painikkeen nimelle.



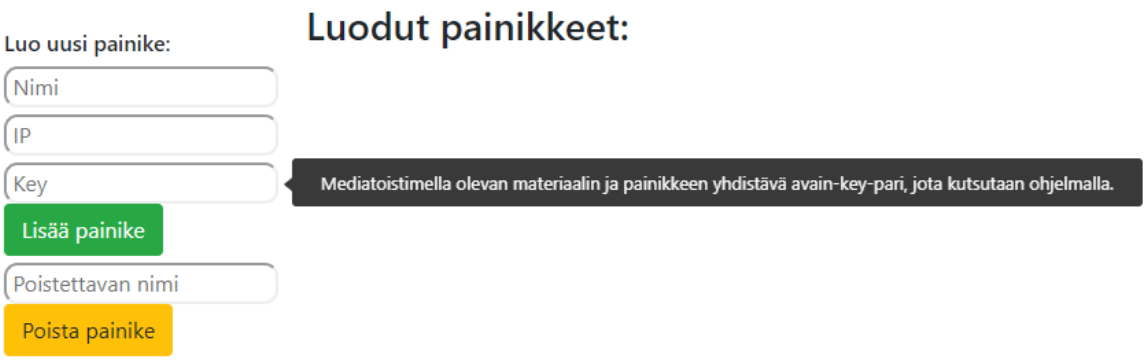
Kuva 5. Nimen lisääminen uudelle painikkeelle

Seuraavaksi lisätään painikkeelle tieto IP-osoitteesta, joka toimii mediatoistimen osoitteena. Vihjekuplassa ilmaistaan, minkä IP-osoitetta pyydetään, ja annetaan malliesimerkki, millainen IP-osoite mediatoistimella voisi esimerkiksi olla.



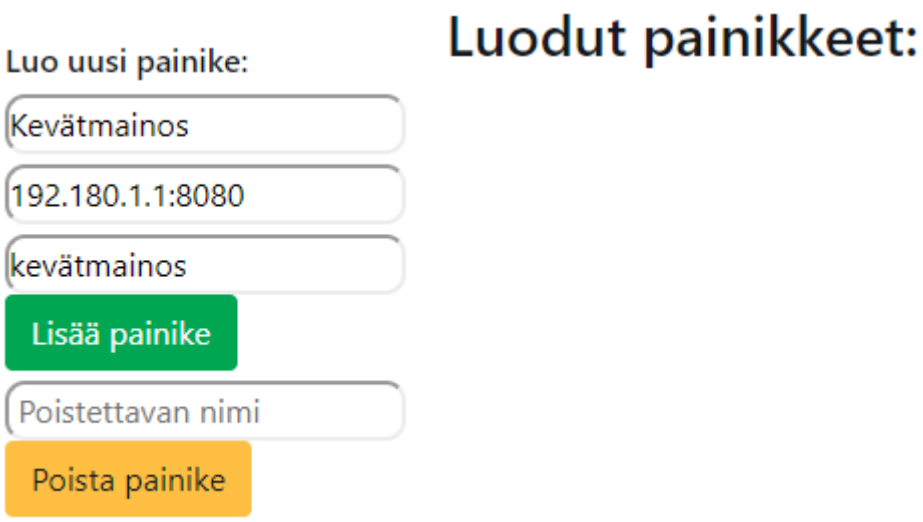
Kuva 6. IP-osoitteen lisääminen uudelle painikkeelle

Uuden painikkeen luonnin Key-kohdassa pyydetään käyttäjää määrittelemään painikkeen Key-arvoksi saman, kuin mediatoistimella olevalle materiaalille asetetun. Mikäli mediatoistimen ja ohjelman avain-key-parit eroavat toisistaan, mediatoistin ei osaa yhdistää oikeaa materiaalia ohjelmalta tulevaan pyyntöön.



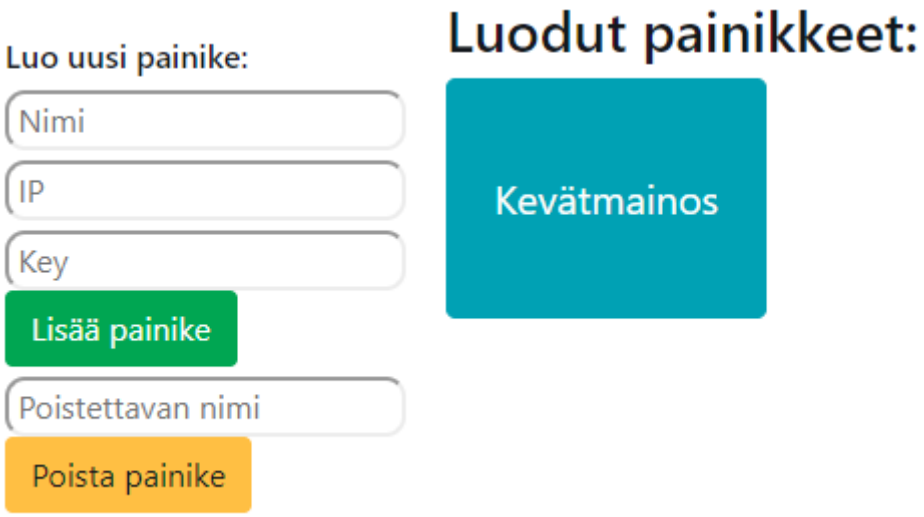
Kuva 7. Keyn lisääminen uudelle painikkeelle

Vihjeiden mukaan luodaan esimerkkinä ”kevätmainos”. Painikkeen nimeksi täytetään mainosta kuvaava ja persoonallinen nimi Kevätmainos. Vastaanottavan esimerkkimediatoistimen IP-osoite; 192.180.1.1:8080 tuodaan julki. Key-kohtaan asetetaan mediatoistimella olevaa materiaalia vastaava avain kevätmainos.



Kuva 8. Esimerkki painikkeen tiedoista

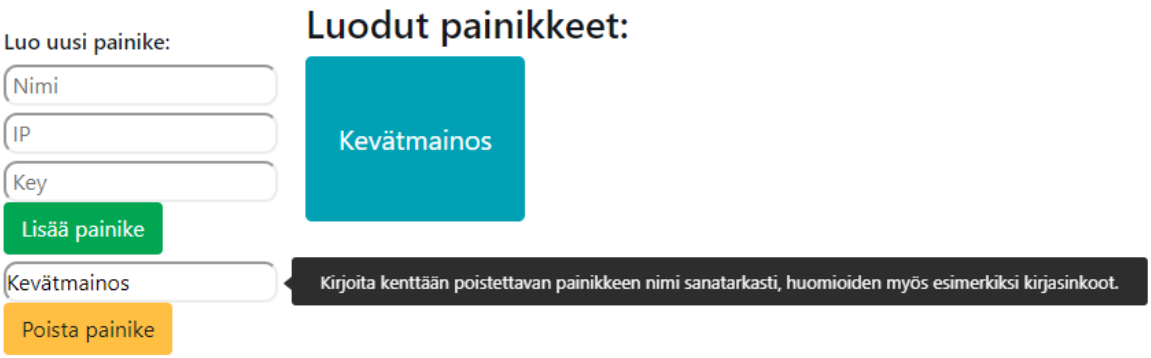
Painettaessa ”Lisää painike”-painiketta, painikkeen tiedot tallentuvat tietokantaan ja muodostuu Kevätmainos-painike ”Luodut painikkeet”-otsakkeen alle. Lomakekentät tyhjentyvät valmiiksi seuraavan painikkeen luontia varten.



Kuva 9. Luotu painike

5.6.2 Painikkeen poistaminen ohjelmasta

Painikkeen poistaminen ohjelmasta tapahtuu selkeällä toimintaperiaatteella. Poistettavan painikkeen nimi kirjoitetaan sanatarkasti kirjasinkoot huomioiden ”Poista painike”-painikkeen yläpuolella olevaan lomakekenttään. Vihjekuplassa tarjotaan vielä lisäselvennystä poistoon. Painettaessa ”Poista painike”-painiketta painike poistetaan tietokannasta ja ohjelman käyttöliittymästä.



Kuva 10. Painikkeen poisto

5.7 Projektissa ilmenneet ongelmat

Projektin etenemisessä suurimpia ongelmia aiheutti työn tekijän kokemattomuus valittujen työkalujen käytössä. Erityisesti ReactJS aiheutti pohtimista työn tekijällä, sillä vaikka jonkin työvaiheen olisikin osannut toteuttaa normaalilla JavaScript-, HTML- ja CSS-kombinaatioilla, toi ReactJS omat haasteensa työvaiheen toteuttamiseen sen vaatiessa omanlaisensa ajattelutavan toteuttaa asioita. ReactJS-kirjaston moduulit toivat myös omat haasteensa vähän kokemusta omaavalle tekijälle.

6 YHTEENVETO JA POHDINTA

Kehittämistyössä oli tavoitteena luoda ja kehittää alkuperäisen jo olemassa olevan ohjelman pohjalta päivitetty versio digitaalisten infonäyttöpäätteiden ohjaamiseen. Jo olemassa olevan ohjelman ollessa ohjelmoitu käyttäen JavaScript-, HTML- ja CSS-kombinaatioita, päätettiin hyödyntää modernimpaa kehitystyökalua, ReactJS-kirjastoa. Kehittämistyön tekijällä oli jo jonkin verran kokemusta web-ohjelman kehittämisestä aikaisemman jo olemassa olevan ohjelman ohjelmoinnin myötä.

Projektityö koostui pitkälti ohjelmointiin liittyvien toimintatapojen selvittelystä. Projektin edetessä kohdattiin monia uusia asioita, joiden selvittämiseen kului aikaa. Myös ohjelman suunnittelu vei aikaa, mutta se oli opettavaista ja kehittävää.

Kehitystarpeita kartoitettaessa tultiin siihen tulokseen, että alkuperäiseen verrattuna päivitetystä versioista oli tultava huomattavasti helppokäyttöisempi, jotta laajempi käyttäjäkunta voisi sitä käyttää ilman suurempia tietoteknisiä taitoja. Helppokäyttöisyyttä tavoiteltaessa kirjallisuudesta löytyi useita esimerkkejä ja asioita, jotka olisi hyvä ottaa huomioon ohjelman käyttöliittymää suunniteltaessa.

Suurin heikkous jo olemassa olevassa ohjelmassa oli ominaisuuspuute, jolla voisi lisätä ja poistaa painikkeita ohjelmaan muutoin kuin ohjelmakoodia muuttamalla. Kehityskohteena olleet ominaisuuspuutteet saatiin lisättyä ohjelmaan, joten yritys kehittää ohjelmaa onnistui siltä osin. Kehitysyritystä voi luonnehtia onnistuneeksi tärkeimpien puuttuneiden ominaisuuksien lisäämisen myötä. Kehitystarpeita ohjelmalle löytynee ulkonäöstä sen ollessa hieman askeettinen. Lisäkehitystarpeisiin kuulunee myös painikkeiden jäsentelyyn liittyviä tekijöitä. Luotaessa käyttöliittymää suurelle joukolle erinäisiä painikkeita olisi varmastikin kohtuullista jaotella painikkeita esimerkiksi alaotsikoiden alle. Mahdollisuus käyttää käyttöliittymää mobiililaitteelta voisi olla mahdollinen kehityskohde tai jopa kehitystarve.

Projektissa suurimpia pohdiskelun aiheita työn tekijälle tuotti ReactJS ja siihen liittyvien asioiden ja työkalujen kanssa toimiminen. Suurin osa ongelmista johtui työn tekijän kokemattomuudesta työkalujen kanssa. Ongelmista kuitenkin päästiin yli, kunhan niihin vain uhrattiin aikaa.

Opinnäytetyön aiheena olevan kehittämistyön tekemisestä koen olleen suurta hyötyä työn tekijälle. Projekti kartutti ohjelmointikokemusta yleisellä tasolla, sekä harjoitti modernien ohjelmointityökalujen ja -tapojen käyttöä. Käyttöliittymäsuunnittelun teoreettisesta puolesta saatu hyöty on merkittävä.

LÄHTEET

Agiliq. 25.5.2018. Understanding State and Props in ReactJS. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://www.agiliq.com/blog/2018/05/understanding-react-state-and-props/

Bits and Pieces. 22.8.2018. 1Understanding React v16.4+ New Component Lifecycle Methods. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://blog.bitsrc.io/understanding-react-v16-4-new-component-lifecycle-methods-fa7b224efd7d

Chinnathambi, K. 2017. Learning React. Pearson Education, Inc.

DigitalSignage. 2.9.2018. Mikä on Digital Signage?. [Verkkosivusto]. [Viitattu 17.2.2019]. Saatavana: http://www.digitalsignage.fi/sivu/fi/digitalsignage/mika_on_digital_signage/

Github. 2.7.2013. Releases. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://github.com/facebook/react/releases?after=v0.4.0

Horton, A., Vice, R. 2016. Mastering React. [E-kirja]. Packt Publishing. [Viitattu 16.4.2019]. Saatavissa: https://web.b.ebscohost.com/ehost/ebookviewer/ebook/bmxlYmtfXzExODM5ODhfX0FO0?sid=f521703d-1d9a-4156-8973-3bab4c287271@pdc-v-sessmgr05&vid=0&format=EB&rid=1

jQuery. Ei päiväystä. What is jQuery?. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://jquery.com/

Ledimedia Oy. Ei päiväystä. Digiaikaista näkyvyyttä. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://ledimedia.fi/

Medium. 2.9.2018. What is Digital Signage?. [Verkkosivusto]. [Viitattu 17.2.2019]. Saatavana: https://medium.com/@wiredstore/what-is-digital-signage-b6ab58ed02a8

Medium. 9.6.2016. Top 32 Sites Built With ReactJS. [Verkkosivusto]. [Viitattu 16.4.2019]. Saatavana: https://medium.com/@coderacademy/32-sites-built-with-reactjs-172e3a4bed81

Medium. Ei päiväystä. Components and Props. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://reactjs.org/docs/components-and-props.html

Nielsen Norman Group. 22.8.1999. Do Interface Standards Stifle Design Creativity?. [Verkkosivusto]. [Viitattu 25.2.2019]. Saatavana:

https://www.nngroup.com/articles/do-interface-standards-stifle-design-creativity/

Nielsen Norman Group. 24.4.1994. 10 Usability Heuristics for User Interface Design. [Verkkosivusto]. [Viitattu 25.2.2019]. Saatavana: https://www.nngroup.com/articles/do-interface-standards-stifle-design-creativity/

Nodejs. Ei päiväystä. Introduction to Node.js. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://nodejs.dev/introduction-to-nodejs

Nodejs. Ei päiväystä. Run Node.js scripts from the command line. [Verkkosivusto]. [Viitattu 16.4.2019]. Saatavana: https://nodejs.dev/run-nodejs-scripts-from-the-command-line

React. Ei päiväystä. Introducing JSX. [Verkkosivusto]. [Viitattu 18.4.2019]. Saatavana: https://reactjs.org/docs/introducing-jsx.html

React. Ei päiväystä. React. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://reactjs.org/

React. Ei päiväystä. State and Lifecycle. [Verkkosivusto]. [Viitattu 12.4.2019]. Saatavana: https://reactjs.org/docs/state-and-lifecycle.html

Sinkkonen, I., Nuutila, E. & Törmä, S. 2009. Helppokäyttöisen verkkopalvelun suunnittelu. Helsinki: Tietosanoma Oy

Sinopoli, J. 2010. Smart Building Systems for Architects, Owners, and Builders. [Verkkokirja]. Knovel-tietokanta [Viitattu 16.3.2019]. Saatavissa: https://app.knovel.com/web/view/khtml/show.v/rcid:kpSBSAOB01/cid:kt009F9GHJ/viewerType:khtml//root_slug:smart-building-systems/url_slug:digital-signage?b-q=digital%20signage&sort_on=default&b-subscription=true&b-group-by=true&page=7&b-sort-on=default&b-content-type=all_references&include_synonyms=yes&view=collapsed&zoom=1&q=digital%20signage

SQLite. Ei päiväystä. About SQLite. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://www.sqlite.org/about.html

w3schools.com. Ei päiväystä. Bootstrap Get Started. [Verkkosivusto]. [Viitattu 15.4.2019]. Saatavana: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

w3schools.com. Ei päiväystä. What is React?. [Verkkosivusto]. [Viitattu 11.4.2019]. Saatavana: https://www.w3schools.com/whatis/whatis_react.asp