SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

DIALOGIEN JA KYSELYIDEN UPOTTAMINEN WWW-SI-VUILLE

Sovelluksen sekä hallintatyökalun toteutus

TEKIJÄ/T: Tomi Tolvanen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala			
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma			
Työn tekijä(t) Tomi Tolvanen			
Työn nimi Dialogien ja kyselyiden upottaminen www-sivuille – sovelluksen sekä hallintatyökalun toteutus			
Päiväys	19.5.2020	Sivumäärä/Liitteet	23
Toimeksiantaja/Yhteistyökumppani(t) Qibbie Mobile Oy			

Tiivistelmä

Opinnäytetyön aiheena oli luoda EverBetter-alustalle työkalu, joka mahdollistaa älykkäiden dialogien ja perinteisten kyselyiden upottamisen helposti ulkopuolisille verkkosivustoille. Lisäksi tavoitteena oli luoda alustalle kokonaan uusi, ”nopea palaute”-tyylinen kyselytyyppi, joka on myös mahdollista upottaa. Tärkeä osa työtä oli automaattisesti luotavan raportin suunnittelu ja kehitys alustalle.

Työn tärkeimpiä vaiheita itse upotuksen luomisen ohella oli selvittää, miten muille sivuistoille olisi mahdollista upottaa kokonaisia React-sovelluksia ilman, että se vaikuttaa esimerkiksi sivuston tyylittelyihin tai käytettävyyteen. Toinen tärkeä vaihe oli käytettävän sivuston varmentamisen suunnittelu, jotta upotusta ei voida käyttää miltä tahansa sivustolta.

Front-end toteutettiin käyttämällä JavaSciprtin React-käyttöliittymäkirjastoa. Rajapinta ja back-end toteutettin Java-ohjelmointikielellä Play-viitekehystä hyödyntäen. Tietokantana käytettiin MongoDB NoSQL-tietokantaa. Työn lopputuloksena syntyi toimiva kokonaisuus: alustalle työkalu upotusten luontia varten, sovellus sekä automaattinen raportointi uudelle kyselytyypille. Tulevaisuudessa upotuksen kehittämistä jatketaan uusilla ominaisuuksilla.

Avainsanat
JavaScript, React, Java

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Tomi Tolvanen			
Title of Thesis Embedding Dialogs and surveys to the webpages – developing the application and control panel			
Date	19 May 2020	Pages/Appendices	23
Client Organisation /Partners Qibbie Mobile Oy			

Abstract

The purpose of the thesis was to create a new feature to the EverBetter platform. The new feature would easily allow to add and embed logical dialogs and more traditional styled surveys to external websites. Another purpose of the thesis was to add a new question type called ”instant feedback”, which would also be possible to be embedded. An important part of the thesis was to design and develop an automatically created report to the platform.

Most important steps of the thesis, alongside of creating the embedded system, were to figure out how to embed whole React applications to external websites without it breaking styles or interfere with the usability of the site and to design the validation of the used site to ensure the embed could be used only on defined websites. The Front-end was implemented by using JavaScript’s React user interface library. Back-end and APIs were implemented using the programming language Java and its Play framework. The database was chosen to be NoSQL database MongoDB.

As a result of the thesis, functioning features were implemented: a tool for creating and modifying embedded codes, a new React application and new question type with automatic reports. In the future the development of the application will continue with new features.

Keywords
JavaScript, React, Java

ESIPUHE

Kiitos Qibbie Mobile Oy:lle opinnäytetyön aiheesta sekä ohjeistuksesta työn aikana. Kiitos lehtori Jussi Koistiselle ohjauksesta.

Kuopiossa 19.5.2020

Tomi Tolvanen

SISÄLLYSLUETTELO

1 JOHDANTO 6

2 TERMIT JA LYHTENTEET 7

3 KÄYTETYT TEKNIIKAT JA KEHITYSYMPÄRISTÖ 8

3.1 Käytetyt tekniikat 8

3.1.1 ReactJS 8

3.1.2 Java 9

3.1.3 Play-viitekehys 9

3.1.4 MongoDB 9

3.1.5 Nginx 10

3.2 Kehitysympäristö 10

3.2.1 Eclipse IDE 10

3.2.2 Robo 3T 10

3.2.3 Visual studio code 11

3.2.4 Github 11

4 TYÖN TOTEUTUS 12

4.1 Feedback kyselytyyppi 13

4.2 Hallitapaneelin työkalu 14

4.3 React-sovellus 16

4.4 Raportointi 18

4.5 Upotten hakuscripti 19

4.6 Palvelun julkaiseminen 20

5 YHTEENVETO 21

1 JOHDANTO

EverBetter on Qibbie Mobile Oy:n kehittämä ja ylläpitämä palvelu, jonka avulla on mahdollista kerätä asiakaspalautteita älykkäiden dialogien (chat-bot-tyylisesti) ja perinteisten kyselyiden avulla. Palvelu luo automaattisesti raportit vastauksien perusteella. Lisäksi palvelussa on mahdollista asettaa tehtäviä, seurata riskejä sekä luoda suunnitelmia.

Opinnäytetyön tavoitteena oli lisätä EverBettter-palveluun kokonaan uusi ominaisuus: mahdollisuus upottaa kyselyitä käyttäjän omille verkkosivustoille. Toinen tavoite oli luoda palveluun kokonaan uusi kysymystyyppi: ”nopea palaute”-tyylinen kysymys, jossa kysymyksiä on vain yksi ja siihen vastataan käyttämällä erilaisia ikoneita, kuten hymynaama tai peukalo ylös/alas. Tarkoituksena oli myös luoda nopealle palautteelle oma raportointi, mistä pystyi seuraamaan vastauksien määrää ja pisteitä sekä miltä sivulta vastaukset oli annettu.

Työkalun tekeminen jakautui kahteen osaan: EverBetter-palvelun käyttöliittymässä oleva työkalu millä voidaan luoda upotukselle skripti ja säätää käytettäviä asetuksia. Toinen osa oli itse upotuksen tekeminen erillisenä React-sovelluksena, joka upotettaisiin käytettävään sivustoon JavaScriptin avulla. Yksi tärkeimmistä ominaisuuksista työtä työtä tehdessä oli helppokäyttöisyys, kun yhteys sivun ja upotuksen välillä on luotu, sitä voidaan päivittää ja muokata suoraan EverBetter-palvelun kautta käyttöliittymästä reaaliaikaisesti.

Työtä toteutusta suunnitellessa kävin esimieheni kanssa läpi erilaisia mahdollisuuksia ja ideoita, miten työtä kannattaisi lähteä toteuttamaan. Toteutusvaiheessa työskentelin pääsääntöisesti yksin ja pidimme tarvittaessa esimieheni kanssa palavereita työn edistymisestä ja seuraavista vaiheista.

2 TERMIT JA LYHTENTEET

Java on Sun Miscrosystemssin kehittämä ohjelmointikieli

Play Framework on Scala-ohjelmointikielellä tehty viitekehys, jota voidaan käyttää kähitykseen

Kirjasto on kokoelma aliohjelmia tai luokkia, joita käytetään ohjelmistojen kehittämiseen

JavaScript on web-ympäristössä käytettävä komentosarjakieli, jolla voidaan luoda dynaamista sisältöä sivustoille

ReactJS on Facebookin kehittämä JavaScript-kirjasto, jota voidaan käyttää käyttöliittymien rakentamiseen

REST-rajapinta (Representational State Transfer) on HTTP-protokolaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteutukseen

3 KÄYTETYT TEKNIIKAT JA KEHITYSYMPÄRISTÖ

Tässä osiossa käydään läpi opinnäytetyön teon aikana käytetyt tekniikat sekä kehitysympäristö. Tekniikoiden valinta oli jo heti projektin alkuvaiheessa selvä, sillä aikaisempi kokemus työn puolesta vaikutti merkitsevästi valittuihin tekniikoihin. Kehitysympäristön valita perustui myös jo entuudestaan tuttuihin työkaluihin työn puolesta.

3.1 Käytetyt tekniikat

Tässä osiossa käydään opinnäytetön käytetyt tekniikat. Back-endinä toimi EverBetter-palvelun jo olemassa oleva Play-viitekehyksellä tehty REST-rajapinta, johon lisättiin tarvittavat luokat sekä funktiot. Tietokantana toimi NoSQL tietokanta MongoDB. Front-end toteutettiin käyttämällä Facebookin kehittämää ReactJS JavaScript-viitekehystä.

3.1.1 ReactJS

React on Facebookin kehittämä ja ylläpitämä komponenttipohjainen JavaScript-kirjasto käyttöliittymien luomiseen (React).

Yksi Reactin vahvuuksista on sen nopeus: tiedon muuttuessa, React päivittää vain ne komponentit, joissa tieto muuttuu ilman että tarvitsee päivittää koko sivua. Käyttöliitttymien kehittäminen on nopeaa ja vaivatonta, sillä kun ohjelmakoodia muutetaan ja tallennetaan, se näkyy heti selaimessa ilman viivettä.

```
class HelloWorld extends React.Component {
  constructor() {
    super();
    this.state = {
        name: "Tomi"
    }
  }

  render() {
    return (
    <h1>Hello, my name is {this.state.name}</h1>
    )
  }
}
```

KUVA 1. Esimerkkikuva yksinkertaisesta komponentista

Opinnäytetyössä kaikki käyttöliittymät tehtiin Reactilla. Upotteiden hallitsemiseen ja raporttien tarkkailuun tehtiin Everbetter-alustalle uudet komponentit, mutta sovellukselle joissa upotuksia käytettäisiin luotiin kokonaan uusi React-sovellus.

3.1.2 Java

Java on Microsystemsin kehittämä ja julkaisema olio-ohjelmointikieli. Java julkaistiin alun perin vuonna 1995. Oliorakenteen ansiosta Java-ohjlemia on helppo tehdä ja laajentaa. Javan hyvinä puolina voidaan mainia mm. usean tehtävän samanaikainen suorittaminen (multi-threading), yksinkertaisuus ja helppous oppia sekä mahdollisuus käyttää Javaa useilla eri alustoilla (Java tutorial, tutorialspoint).

Opinnäytetyössä rajapinta toteutettiin käyttäen Javaa tukevaa Play-viitekehyksestä.

3.1.3 Play-viitekehys

Play on avoimen lähdekoodin viitekehys, joka on tehty Scala-ohjelmointikielellä. Playtä voidaan käyttää myös muilla ohjelmointikielillä, jotka kääntyvät JVM (Java Virtual Machine) bittikoodiksi, kuten tämän työn tapauksessa, Javaksi. Playllä on mahdollista tehdä kokonaisia internetsovelluksia käyttäen malli-näkymä-käsittelijä (MVC, model-view-controller) ohjelmistoarkkitehtuuria, mutta tässä opinnäytetyössä sitä käytettiin vain REST-rajapintana.

Opinnäytetyössä valitsin Play-viitekehyksen rajapinnaksi, koska EverBetterissä käytettävä rajapinta on jo valmiiksi toteutettu Javalla Play-viitekehystä käyttäen.

3.1.4 MongoDB

MongoDB on yleiskäyttöinen, asiakirjapohjainen hajautettu tietokanta, joka on rakennettu nykyaikaisille sovelluskehittäjille ja pilvipohjaisten palveluiden aikakaudelle (MongoDB).

MongoDB on avoimeen lähdekoodiin perustuva asiakirjapohjainen no-sql tietokanta. Sitä on helppo ja nopea oppia käyttämään.

Relaatiokantoihin verrattuna MongoDB:n suurimmat vahvuudet on parempi skaalautuvuus tiedon määrien kasvaessa ja tiedon tallennuksessa tiedot voidaan tallentaa suoraan JSON-muodossa tietokantaan, eikä tarvitse huolehtia tiedon muuttamista relaatiomuotoon. Tämä helpottaa tiedon tallennusta olio-ohjelmoinnissa. Yhtenä vahvuutena voidaan myös pitää tietokannan käsittelyn helppoutta (SQL vs. NoSQL, xplenty).

3.1.5 Nginx

Nginx on Igor Sysojevin kehittämä avoimeen lähdekoodiin perustuva WWW- ja proxy- palvelin. Se on julkaistu ensimmäisen kerran vuonna 2004. Nginx on suosituin käytetty palvelin, Netcraft arvioi, että lähes 38% kaikista aktiivisista nettisivuista käyttäisi Nginxiä (January 2020 Web Server Survey, Netcraft).

3.2 Kehitysympäristö

Tässä osiossa käydään läpi työssä käyttämäni kehitysympäristö. Käytin työssä back-endin luomista ja muokkaamista varten Eclipse IDE:tä, tietokannan tarkastelua varten käytin Robo 3T:tä sekä käyttöliittymiä varten Visual Studio Codea.

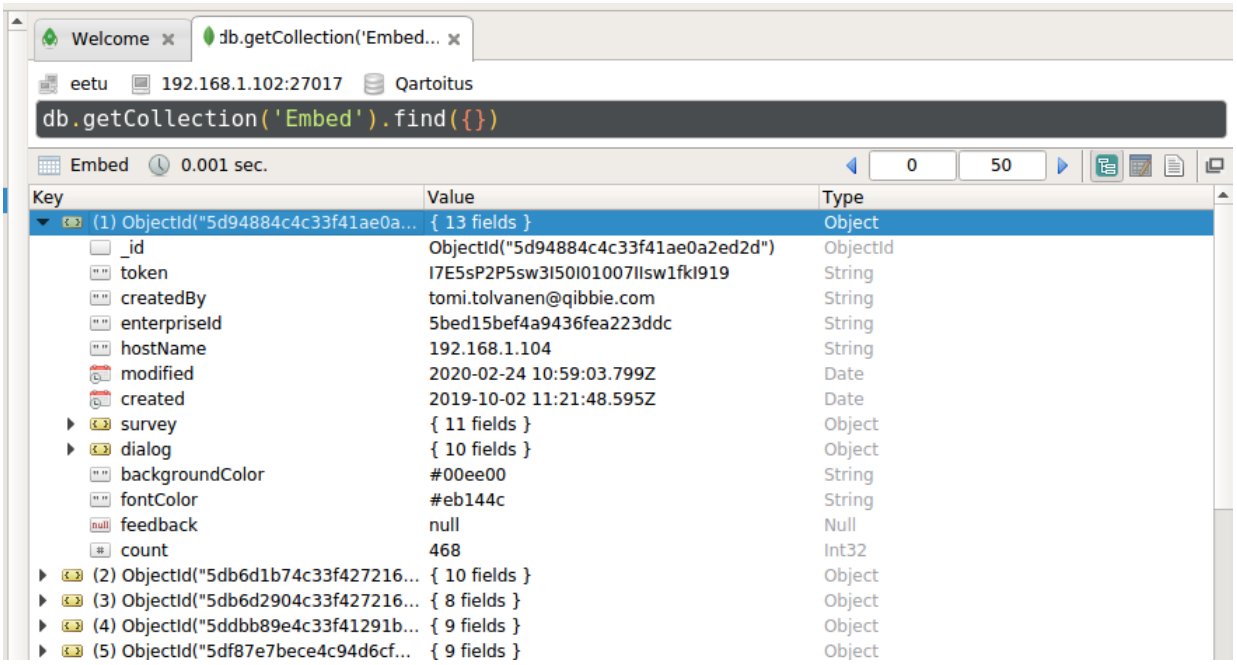
3.2.1 Eclipse IDE

Eclipsen kehityksen aloitti IBM vuonna 1993 ja vuonna 2001 se julkaistiin avoimelle lähdekooodille. Vuodesta 2004 lähtien Eclipseä on kehittänyt Eclipse Foundation säätiö. Eclipse on kehitysympäristö, joka tukee muun muassa Javaa ja C-ohjelmointikieliä (Eclipse).

Opinnäytetyössä back-endin kehitys tapahtui Eclipsellä, jonka valitsin aijemman kokemuksen perusteella sekä sen hyvien ominaisuuksien takia.

3.2.2 Robo 3T

Robo 3T on avoimeen lähdekoodiin perustuva graafinen käyttöliittymä MongoDB-tietokannalle. Se on nopea ja helppokäyttöinen, eikä vaadi erillisiä konfiguraatioita toimiakseen. Graafisen käyttöliittymän käyttäminen mahdollisti tiedon tarkastamisen ja poistamisen nopeammin sekä erilaisten tietokantakutsujen testauksen.



KUVA 2. Robo 3T käyttöliittymä

Robo 3T on yksi suosituimpia graafisia käyttöliittymiä MongoDb:lle ja se on edelleen yksi suosituimmista GitHub- projekteista. (Which Is the Best MongoDB GUI, Dzone)

3.2.3 Visual studio code

Visual Studio Code (VSCode) on Microsoftin kehittämä moderni tekstieditori. Vuonna 2019 VSCode oli yksi suosituimmista tekstieditoreista ohjelmoijien keskuudessa (Developer Survey Results, Stackoverflow) VSCodelle on saatavissa lukuisia lisäosia ja sitä on mahdollista muokata omiin tarpeisiinsa todella hyvin lukuisien asetuksien ja teemojen ansiosta. Lisäksi sille on tarjolla useita hyödyllisiä lisäosia, jotka nopeuttavat ja helpottavat kehitystä.

Valitsin front-endille kehitystuokaluksi juuri VSCoden, koska lukuisat pikakomennot, kuten koodin automaattinen muotoilu, automaattisten luokkien ja kirjastojen tuonti, helpottivat ja nopeuttivat kehitystä. Lisäksi siinä on sisäänrakennettuna valmius käyttää GitHub versionhallintaa

3.2.4 Github

Githubilla voidaan hallita ja säilyttää projektien versioita. Vaikka Github on useimmiten käytetty koodin, sitä voidaan käyttää myös minkä tahanasa tiedostomuodon, kuten Word-tiedostojen tai Final Cut projektien tallennukseen ja hallintaan. Githubia voidaan ajatella arkistointijärjestelmänä, jossa dokumentin kaikki versiot ovat tallessa (What exactly is Github anyway, Techcrunch).

Opinnäytetyön versionhallinnaksi valitsin Githubin, koska minulla aikaisempaa kokemusta sen käytöstä sekä se on käytössä yleisesti jokaiselle projektille työpaikalla.

4 TYÖN TOTEUTUS

Opinnäytetyön tärkeimmät kohdat olivat tehdä työkalu, jonka avulla on mahdollista upottaa älykkäitä chatbot-tyylisiä dialogeja ja perinteisiä kyselyitä ulkoisille nettisivuille helposti. Lisäksi EverBetter-palveluun oli tarkoitus tehdä uusi kyselytyyppi: nopea palaute, jonka avulla voidaan kerätä nopeita, yhden kysymyksen, palautteita käyttäjältä. Nopeasta palautteesta luodaan automaattisesti päivittyvä raportti EverBetter-palveluun.

Tavoitteena oli tehdä mahdollisimman helppokäyttöinen ja yksinkertainen työkalu, joka mahdollistaa upotuksien tekemisen. Työkalulle luotiin jo olemassa olevaan React-sovellukseen uudet komponentit, joiden avulla käyttäjä voi luoda ja hallita upotuksia. Upotukselle päätettiin luoda oma React-sovellus, joka haettaisiin palvelimelta ja liitettäisiin sivustoon käyttämällä JavaScriptia.

Työn suunnittelu aloitettiin määrittelemällä työn tavoitteet ja suunniteltiin alustavaa ulkoasua sekä miten ja millä työ toteutetaan. Ensimmäisenä työssä luotiin feedback kyselytyyppi ja ensimmäinen versio hallintapaneelin työkalusta, jolla pystyi aluksi vain luomaan uuden upotuksen.
Seuraava vaihe oli tutkia, miten React-sovelluksen upottaminen on mahdollista ulkopuoliselle sivustolle JavaScriptin avulla.

Kun uusia upotuksia pystyi luomaan ja React-sovellukselle oli valmis pohja tehtynä, siirryttiin toteuttamaan ominaisuuksia ja toiminnallisuuksia seuraavassa järjestyksessä:

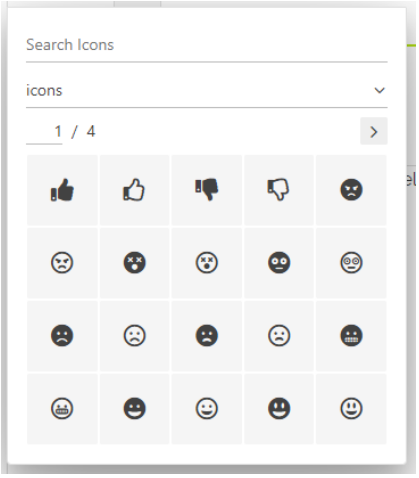
1. Kyselyt
    a. Kaikki kolme kyselytyyppiä on ensin kiinni ja ne pystytään avaamaan
    b. Vastauksien tallentaminen tietokantaan
    c. Dialogille ja kartoitukselle kaikki kysymystyypit
2. Feedback
    a. Vastauksen jälkeen käyttäjälle loppupalautteen näyttäminen ja feedbackin automaattinen sulkeutuminen hetken kuluttua.
    b. Sivuston polun lisääminen vastaamisen yhteydessä
3. Hallintapaneeli
    a. Automaattinen koodin ja kopioitavan skriptin luonti uudelle upotukselle
    b. Kyselyiden valinta
    c. Mahdollisuus muokata ja tyylitellä tekstejä
    d. Upotteen sijainnin valinta
4. Raportointi
    a. Feedback kysymyksen keskiarvon laskeminen
    b. Polkujen ja vastauksien näyttäminen eri poluista
5. Upotteen hakuscripti
    a. Upotteen mukana tulevan koodin tarkistaminen
    b. Upotteessa käytettävien kyselyiden aktiivisuuden tarkistaminen

   c. Domainin tarkistaminen vastaamaan määriteltyä domainia
6. Viimeistely
   a. Kaikkien kyselyiden animointi
   b. Kyselyiden piilottaminen vastauksen tai sulkemisen jälkeen
   c. Mahdollisuus jatkaa kyselyitä
7. Julkaiseminen
   a. Skripti, jonka avulla tehdään React-sovelluksesta tuotantoversio ja ladataan palvelimelle

4.1 Feedback kyselytyyppi

EverBetter-alustalla oli valmiina jo kaksi kyselytyyppiä: tavallisen kyselyn tyylinen kartoitus sekä keskustelu-bot tyylinen dialogi.

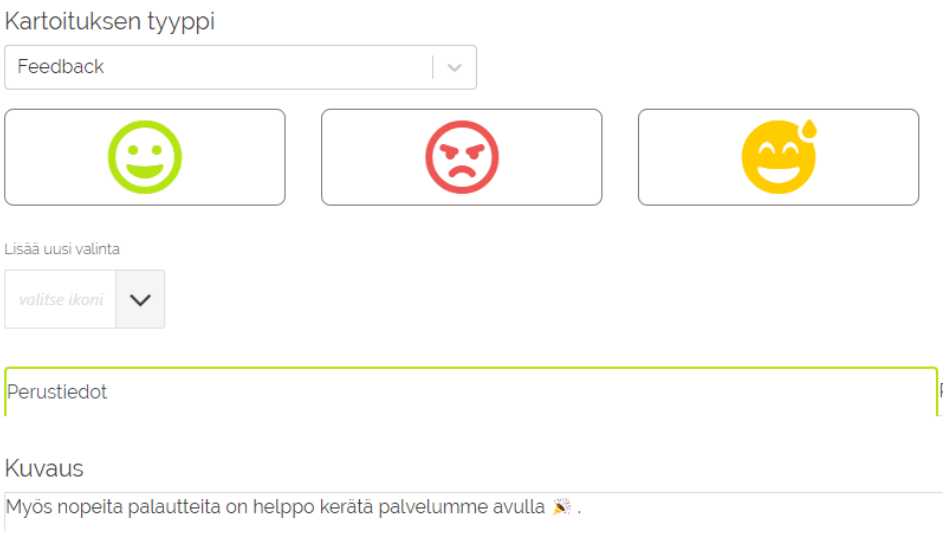
Osana opinnäytetyötä haluttiin tehdä alustalle uuden tyylinen kysymystyyppi, joka toimisi erityisen hyvin upotteena. Feedback on nopeasti kerättävä palaute, jossa on vain yksi kysymys ja vastausvaihtoehdot ovat ikoneita, joita voi kyselyä luodessa valita 1-x määrän (kuva 3).



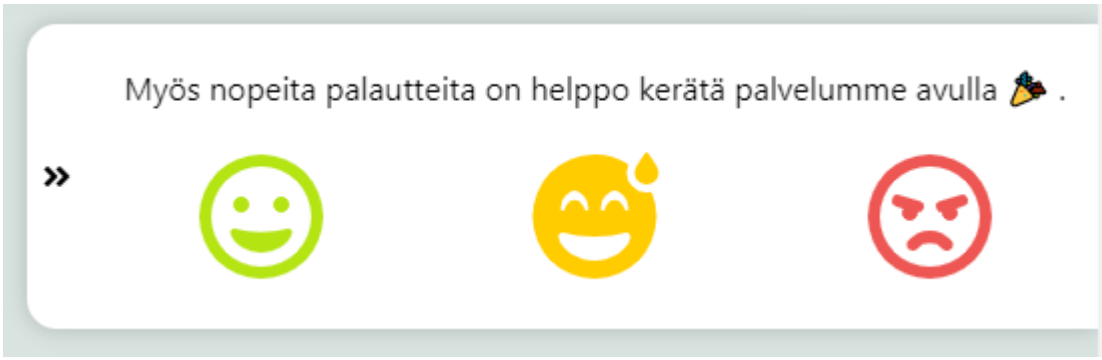
KUVA 3. Erilaisia valittavia ikoneita

Alustalle tehtiin mahdollisuus tehdä feedback-kyseyitä, joihin voi valita haluamansa kysmystekstin, palautteen sekä valita käytettävät ikonit ja niiden värit useista erinlaisista ikoneista. Jokaiselle vastausvaihtoehdolle annetaan pisteet, joiden perusteella raportissa lasketaan kyselyn tulos vastauksien keskiarvosta.

Vastattaessa kysymykseen, vastaajalle näytetään kiitosteksti, jota voidaan tyylitellä tai siihen voidaan lisätä kuvia tai videoita. Lisäksi on mahdollisuus lisätä tieto, kuinka monta prosenttia vastaajista on vastannut samalla tavalla kuin vastaaja.



KUVA 4. Kysymyksen asetuksia



KUVA 5. Esimerkki feedback kysymyksestä upotettuna

4.2 Hallitapaneelin työkalu

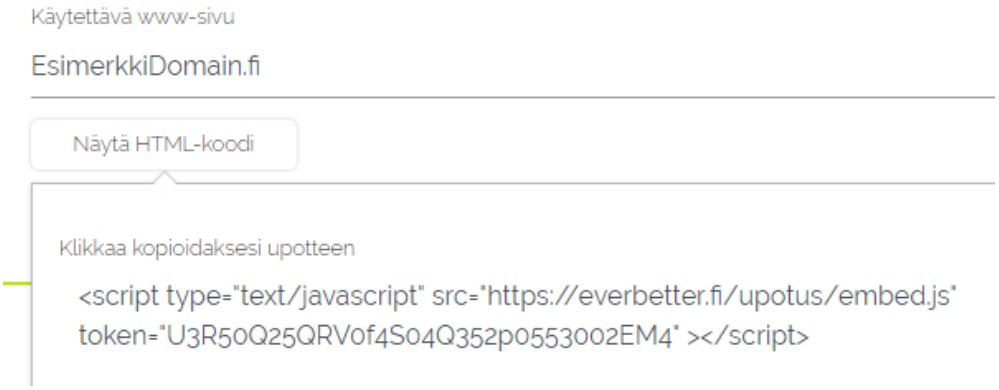
Uutena ominaisuutena hallintapaneeliin tehtiin uusi työkalu, mikä mahdollistaa asiakkaiden kyselyiden upottamisen omille nettisivuille. Käyttäjä valitsee millä domainilla upotus toimii. Domainin valinta tehtiin palveluun koska haluttiin, että upotusta ei ole mahdollista laittaa mille tahansa sivustolle ja siihen voidaan vastata vain määritetyltä sivustolta.

Domainin asettamisen jälkeen käyttäjä voi valita, minkä tyyppisiä kyselyitä hän haluaa lisätä sivulle. Linkitetyllä sivustolla voi olla kaikki kolme kyselytyyppiä (kartoitus, dialog, feedback) aktiivisena, jos käyttäjä niin haluaa. Dialogille ja kartoitukselle tehtiin mahdollisuus sijoittaa se joko keskelle sivua pop-up tyylisesti tai sivuston vasempaan tai oikeaan alareunaan (kuva 6).
Hallintapaneeliin lisättiin laskuri, josta näkee montako kertaa upotus on ladattu onnistuneesti.

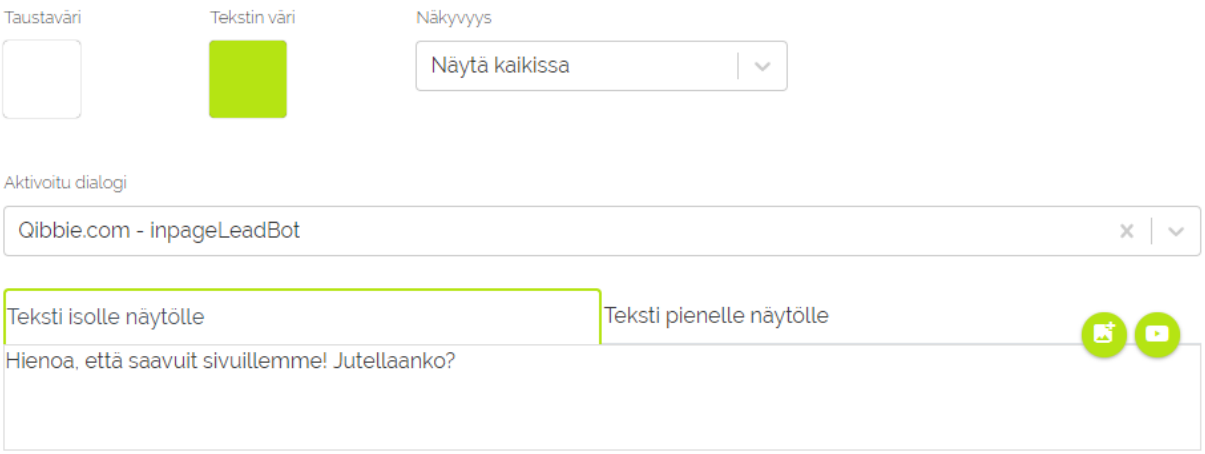
KUVA 6. Upotteen näkyvyyden asetus säädetty näkymään vasemmassa alareunassa.

Dialogissa ja kartoituksessa tehtiin mahdollisuus lisätä teksti, joka näkyy sivulla, kun kysely on suljettu. Lisäksi tehtiin mahdollisuus lisätä erillinen teksti pienemmille (puhelimet, tabletit yms.) ja isommille näytöille. Tekstiä voi tyylitellä ja siihen voi lisätä kuvia, videoita sekä emojeita.

Kaikille kyselytyypeille tehtiin valinta, onko kysely piilotettu puhelimen tai tabletin kokoisilla näytöillä. Käyttäjä voi valita esimerkiksi, että isomalla näytöllä näkyy sekä dialogi ja karoitus kun taas puhelimella näkyy pelkkä dialogi.



KUVA 7. Domain ja valmiiksi generoitu skripti, voidaan kopioida suoraan oman sivuston Html-koodiin

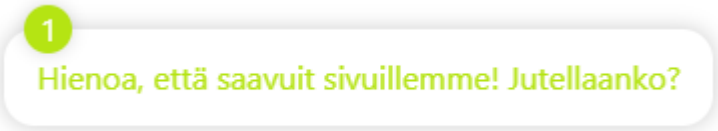


KUVA 8. Dialogin asetuksia

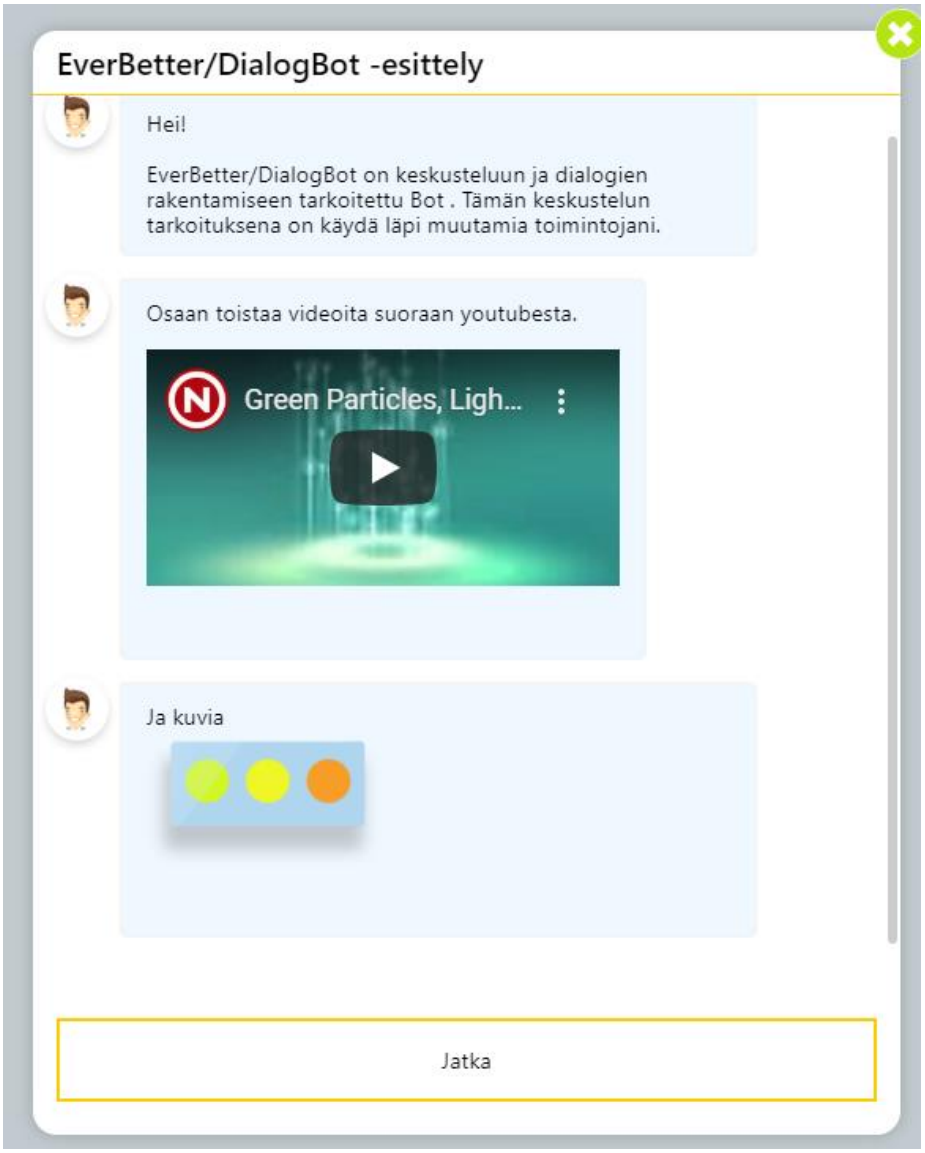
Tavoitteena oli tehdä upotuksesta mahdollisimman helppokäyttöinen ja helppo päivittää ilman, että käyttäjän tarvitsee liittää uusi koodi sivuille jokaisen muutoksen jälkeen. Työkalu suunniteltiin toimimaan niin että kun linkitys sivun ja upotuksen välillä on kerran luotu, kaikki muutokset voidaan tehdä työkalusta käsin ja kaikki muutokset päivittyvät heti.

4.3 React-sovellus

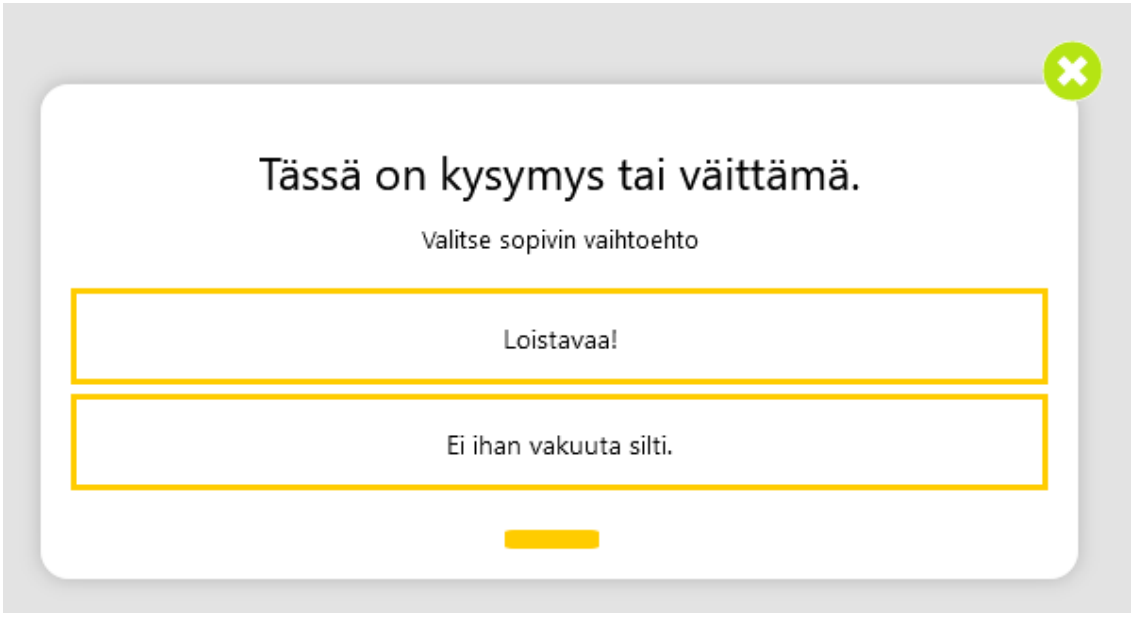
Opinnäytetyölle tehtiin kokonaan uusi React-sovellus, jonka tarkoitus on ainoastaan toimia kyselyiden täyttämistä varten sivustolla, jonne se on upotettu. Ulkoasultaan kyselyt haluttiin tehdä muistuttamaan EverBetterin kotisivuilla tehtäviä kyselyitä, mutta koska tarkoituksena oli upottaa niitä muihin sivuihin, ne eivät voineet olla enää koko sivun kokoisia. Kyselyt ovat myös automaattisesti suljettuna (kuva 9) ja aukeavat vain, jos sivuston vierailija avaa kyselyn (kuva 10 ja 11).



KUVA 9. Sivuston alalaidassa oleva suljettu dialogi



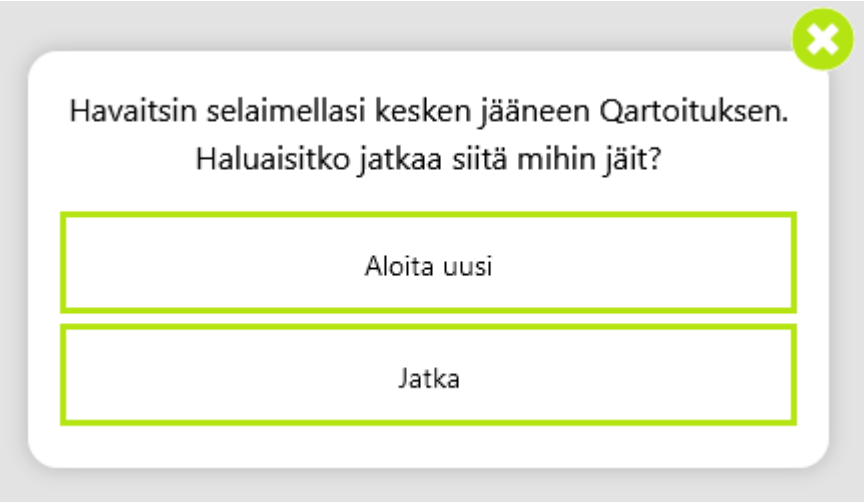
KUVA 10. Esimerkki avatusta dialogista sivun alalaidassa



KUVA 11. Esimerkki avatusta kartoituksesta sivuston alalaidassa

Kyselyn täyttämisen jälkeen merkitään keksiin tieto siitä, että kysely on täytetty eikä se näy käyttäjälle, jos hän palaa sivustolle uudestaan myöhemmin. Käyttäjä voi myös halutessaan piilottaa kyselyt ruksista.

Käyttäjällä on myös mahdollisuus jatkaa jo kerran aloitettua kyselyä myöhemmin. Mikäli kyselyä avatessa havaitaan, että selaimen muistissa on tallennettuna keskeneräinen, kysely kysytään käyttäjältä, haluaako hän jatkaa kyselyä vai aloittaa kokonaan uuden.



KUVA 12. Kyselyn jatkaminen

Sovelluksen oli tarkoitus toimia monella erilaisella sivustolla rikkomatta sivuston tyylittelyjä tai käytettävyyttä, piti sovelluksessa käytettävän Bootstrap-viitekehyksen antaa muokata vain React-sovelluksessa olevien elementtien tyylittelyjä. Tämä onnistui tekemällä Bootstrapistä oman version LESSCSS-prosessointityökalulla ja asettaa että se voi tyylitellä vain elementtejä, jotka ovat sovelluksessa.

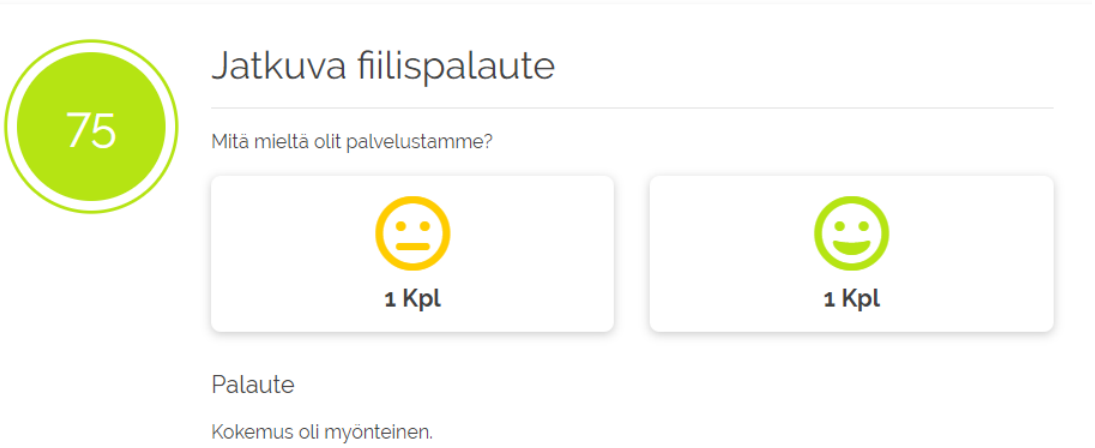
```
▼<div class="embedstrap embed "> == $0
```

KUVA 13. Upotuksessa käytettävä ”embedstrap” luokka. Prosessoitu bootstrap muokkaa vain elementtejä, jotka ovat tämän luokan jälkeen.

Sovellusta voidaan käyttää sivustoissa, jotka ovat tehty myös Reactilla. Sivuston rakenteeseen luodaan oma root-elementti, jonka jälkeen sovelluksen elementit luodaan ja näytetään käyttäjälle.

4.4 Raportointi

Automaattisesti tehtäviin raportteihin lisättiin mahdollisuus tarkastella feedback- kysymysten vastauksien tulosta sekä miltä sivulta vastaus on annettu. Käyttäjä antaa jokaiselle vastausvaihtoehdolle luontivaiheessa pisteet sekä halutessaan palautteen, joiden perusteella raportti luodaan ja päivitetään automaattisesti. Kysymyksen kokonaispistemäärä lasketaan laskemalla kaikkien vastauksien pisteiden keskiarvo.



KUVA 14. Feedbackin pisteet sekä palaute.

Vastauksissa tallennetaan myös verkko-osoitteen polku missä vastaus on tullut. Raportissa voidaan tarkkailla vastauksien määriä eri sivuilta (kuva 15).

KUVA 15. Raportissa näkyvien vastauksien url-osoitteet ja kappalemäärät.

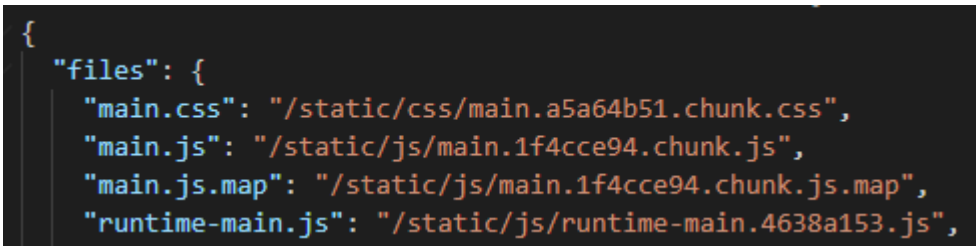
4.5 Upotten hakuscripti

Projektille tehtiin pienempi Javascript-scripti, joka toimii varsinaisen React-sovelluksen ja HTML-sivun välillä. Jokaiselle upotteelle luodaan oma, uniikki, tunniste. Hakuscripti lähettää tunnisteen rajapinnalle, jossa tarkistetaan miltä verkko-osoitteelta pyyntö tuli ja onko se yhteenkuuluva tietokannassa olevaan verkkotunnukseen. Mikäli pyynnön osoite ja verkko-osoite täsmäävät rajapinta palauttaa upotuksen tiedot, jonka jälkeen hakuskripti lataa sivustolle automaattisesti React-sovellukseen kuuluvat JavaScript- ja tyylitiedostot. React- sovellus lataa kyselyt itse.

Jotta React-sovellus voidaan esittää html-sivulla, tarvitsee se ”root”- elementin. Hakuskripti tekee JavaScriptillä div- elementin, jolla on tietty id, jonka avulla React-sovellus voidaan näyttää sivulla.

Suurin haaste hakuscriptin tekemisessä oli että, Käännetyssä React-sovelluksessa JavaScript tiedostojen nimet muutetaan ”hashattuun” muotoon. Tiedostojen nimet muuttuvat joka kerta kun sovellus käännettään tuotantoversioon, joten hakuscriptiin olisi pitänyt muuttaa uudet tiedostojen nimet käsin jokainen kerta. Ongelma saatiin ratkaistua lukemalla tiedostot automaattisesti tehdystä ” asset-

manifest” (kuva 16) nimisestä JSON-tiedostosta mihin kirjoitetaan käännösvaiheessa kaikki tiedostot mitä sovelluksessa on.



KUVA 16. Assert-manifest JSON-tiedostossa löytyvät hashatyt tiedostot JSON objektissa.

4.6 Palvelun julkaiseminen

React-projektin julkaisemisen ja päivittämisen helpottamiseksi tehtiin projektille erillinen skripti, joka kääntää tuotantoversion react-projektista ja lataa sen palvelimelle SSH:n kautta. Palvelimena käytettiin jo olemassa olevaa Linux-palvelinta, missä oli käytössä Nginix. Nginx proxy-palvelimelle luotiin uusi lohko, joka ohjaa kaikki pyynnöt, jossa on polkuna ”/upotus” oikeaan kansioon missä upotteen hakuskripti sijaitsee.

5 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda sovellus, joka olisi mahdollista liittää ulkopuolisille nettisivuille helposti ja vaivattomasti. Lisäksi osana työtä oli luoda upotusten luontia varten käytettävä työkalu sekä uusi kysymystyyppi EverBetter- alustalle. Alustalle tehtiin uusi kysymystyyppi ja siihen liittyvät lisäykset automaattisesti luotaviin raportteihin.

Työn lopputulos vastasi suunnitelmia ja lopputuloksena syntyi toimiva sovellus, josta tuli osa EverBetter-alustaa. Alustalle tehtiin uusi kysymystyyppi sekä työkalu, jolla upotuksia pystyy luomaan ja muokkaamaan. Työ valmistui suunnitellussa aikataulussa. Tulevaisuudessa työtä tullaan jatkokehittämään lisäominaisuuksilla ja toiminnallisuuksilla ja parannuksilla kuten automaatiotestauksien tekeminen ja uudet kysymystyypit.
Tulevaisuudessa on tarkoitus pystyä lisäämään sama upotus monelle eri sivustolle ja laskea niiden tulokset yhteen raportoinnissa.

Sovelluksen kehittämistä jatketaan tulevaisuudessa uusien kysymystyyppien avulla sekä mahdollisesti uusien asetuksien ja säätöjen avulla. Uusia valittavia asetuksia voisivat olla muun muassa uudet kohdat mihin upotuksen voi asettaa, mahdollisuus kytkeä upotteiden animoinnit pois käytöstä kokonaan sekä mahdolliseti upotteen koon säätäminen.

Omasta mielestäni työ onnistui hyvin. Työn aikana toimin pääsääntöisesti itsenäisesti, mutta kävimme esimieheni kanssa läpi tehtyjä asioita sekä mitä tehdään seuraavaksi viikoittain sekä sain tarvittaessa neuvoja ja vinkkejä, miten työtä kannattaisi jatkaa.
Opin työn aikana paljon uutta ja sain syvennystä jo osaamiini asioihin Reactista ja JavaScriptistä. Sain paljon hyvää kokemusta, miten luodaan kokonaan uusia sovelluksia käyttäen Reactia ja mitä kannattaa ottaa huomioon suunnitellessa käyttöliittymää ja sen toiminnallisuutta. Opin myös paljon uusia asioita liittyen rajapintoihin sekä miten tieto liikkuu käyttöliittymän ja rajapinnan välillä sekä miten http-pyyntöjen ylätunnuksia (headers) voidaan hyödyntää ja käyttää.

Jos tekisin jotain toisin työssä niin suunnittelisin React-sovelluksen rakenteen tarkemmin heti alusta lähtien ja suunnittelisin miten eri säädettävät asetukset vaikuttavat sovellukseen. Tämä olisi nopeuttanut tekovaihetta jonkin verran sekä olisi vähentänyt tarvetta tehdä uudelleen joitain tiettyjä asioita käyttöliittymässä.

Huomasin työssäni myös, miten tärketä ja hyödyllistä asioiden suunnittelu ja tutkiminen etukäteen on. Hyvällä suunnnittelulla voidaan välttää useita virheitä sekä koodin uudelleen kirjoittamista koska asiat eivät toimineet halutulla tavalla. Työn aikana ymmärsin myös, miten toimintoja luodessa kannattaa yrittää ottaa huomioon myös mahdolliset tulevaisuuden muutokset. Esimerkiksi lisäominaisuuksia luodessa tai muokatessa jo olemassa olevia ominaisuuksia.

Suurimpina ongelmina työn aikana oli selvittää, miten React-sovelluksen pystyy liittämään ulkopuoliseen sivuun vain yhden liitettävän skriptin avulla tai miten saadaan muokattua vain sovelluksessa olevia Html-elementtejä käyttämällä CSS:ää.

LÄHTEET

Java tutorial, (viitattu 22.2.2020) Saatavissa:
https://www.tutorialspoint.com/java/index.htm

ReactJs, (viitattu 22.2.2020) Saatavissa:
https://reactjs.org/

Eclipse, (viitattu 22.2.2020) Saatavissa:
https://www.eclipse.org/

SQL vs. NoSQL, (viitattu 22.2.2020) Saatavissa:
https://www.xplenty.com/blog/the-sql-vs-nosql-difference/

MongoDB, (viitattu 22.2.2020) Saatavissa:
https://www.mongodb.com/

January 2020 Web Server Survey, (viitattu 22.2.2020) Saatavissa:
https://news.netcraft.com/archives/2020/01/21/january-2020-web-server-survey.html

Developer Survey Results, (viitattu 11.3.2020) Saatavissa:
https://insights.stackoverflow.com/survey/2019#development-environments-and-tools

Which Is the Best MongoDB GUI, (viitattu 23.4.2020) Saatavissa:
https://dzone.com/articles/which-is-the-best-mongodb-gui-2019-update

What exactly is Github anyway, (viitattu 23.4.2020) Saatavissa:
https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/