



OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

KONTTIKORJAAMON WEB- SOVELLUS

Opinnäytetyö

TEKIJÄ: Ragnar Susi

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Ragnar Susi	
Työn nimi Konttikorjaamon web-sovellus	
Päiväys 24.5.2020	Sivumäärä 23 sivua
Ohjaaja(t) Jussi Koistinen, Keijo Kuosmanen	
Toimeksiantaja/Yhteistyökumppani(t) Bertschi Finland Oy	
<p>Tiivistelmä</p> <p>Opinnäytetyön tavoitteena oli suunnitella ja ohjelmoida Bertschi Finland Oy:n konttikorjaamolle web-sovellus. Työn tilaajan tarkoituksena on vaihtaa paperillinen toiminta digitaaliseksi, tabletilla toimivaksi toimintatavaksi. Yhdessä työn tilaajan kanssa lähdettiin suunnittelemaan projektin kannattavuutta. Tavoitteena oli selvittää voisiko korjaamon toimenpiteet hoitaa täysin digitaalisesti, web-sovelluksen avulla ja mitä se vaatisi toimiakseen.</p> <p>Työn tilaajana oli Bertschi Finland Oy, jonka toimintaan kuuluu tarjota kuljetus-, varastointi- ja lisäarvopalveluita petrokemianteollisuudelle. Bertschi Finland Oy:n konttikorjaamon toiminta keskittyy kuljetusyksiköiden kunnossapitoon ja asennustoimintaan, mutta se ei kuulu yrityksen ydinliiketoimintaan. Konttikorjaamon toimintaa tehostettiin kuljetus- ja varastointipalvelujen kysynnän kasvaessa vuonna 2015 investoimalla yrityksen omaan korjaamohalliin.</p> <p>Opinnäytetyötä lähdettiin toteuttamaan jo olemassa olevan web-sovelluksen päälle lisäominaisuutena. Toteutus tehtiin vaiheittain; prosessin aluksi oli tarkoitus muodostaa sovelluksen runko sekä pohtia tärkeimpiä ominaisuuksia tilaajan kannalta. Seuraavassa vaiheessa ohjelmoitiin sovelluksen toiminnot sekä näkymät metadatalle. Sovelluksen perusrakenteen valmistuttua tietokantaan syötettiin korjaamolla käytettävää dataa, joka vietiin web-sovellukseen. Työssä käytettiin ohjelmointikielenä JavaScriptiä ja apuna käytettiin React nimistä kirjastoa. Tietokantana oli NoSQL, jota käytettiin MongoDB alustalla.</p> <p>Opinnäytetyön tuloksena saatiin testikäyttöön soveltuva konttikorjaamon web-sovellus, jota jatkossa aiotaan kehittää lisää. Web-sovellusta on vielä muokattava ja lisäominaisuuksia lisättävä, jotta saadaan mahdollisimman toimiva sovellus, joka vastaa työtilaajan tarpeita parhaimman mukaan.</p>	
Avainsanat konttikorjaamo, web-sovellus, metadata	

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Computer Science			
Author(s) Ragnar Susi			
Title of Thesis Container Repair Web Application			
Date	24 May 2020	Pages	23 pages
Supervisor(s) Mr Jussi Koistinen, Senior Lecturer Keijo Kuosmanen			
Client Organisation /Partners Bertschi Finland Oy			
<p>Abstract</p> <p>The aim of the thesis was to design and program a web application for Bertschi Finland Oy's container repair shop. The purpose of the commissioner was to change the paper operation to a digital, tablet-based operation. The profitability of the project was planned together with the commissioner. The goal was to find out if the workshop's operations could be handled completely digitally, using a web application, and what it would take to operate it. The commissioner of the thesis was Bertschi Finland Oy, whose activities include providing transportation, storage and value-added services to the petrochemical industry.</p> <p>Bertschi Finland Oy's container repair shop focuses on the maintenance and installation of transport units, but it is not part of the company's core business. The operations of the container repair shop were intensified by investing in the company's own workshop hall as the demand for transport and storage services increased in 2015.</p> <p>The implementation was started on top of an existing application as an additional feature. The implementation was done in stages; to start the project, the purpose was to first design the body of the application as well as to consider the most important features for the commissioner. In the next step, the application's functions and views were programmed with metadata. After the basic structure of the application was completed, the data used in the workshop was entered into the database, from where it was imported into the web application. JavaScript was used as the programming language in the project. A library called React was also used as an aid. The database was NoSQL which was used on the MongoDB platform.</p> <p>As a result of the thesis, a web application for a container repair shop suitable for test use was obtained, which will be further developed in the future. The web application still needs to be modified and additional features added to make the application as functional as possible to best meet the needs of the commissioner.</p>			
<p>Keywords</p> <p>Container repair, web-application, metadata</p>			

SISÄLTÖ

1	JOHDANTO	5
2	LYHENTEET JA MÄÄRITELMÄT	6
3	KÄYTETYT TEKNIIKAT	8
3.1	Ohjelmointikieli	8
3.2	Tietokanta	8
3.3	Palvelin	11
3.4	Versionhallinta	11
4	ASIAKAS	12
4.1	Työn tilaaja	12
4.2	Konttikorjaamo	12
5	WEB-SOVELLUS	14
5.1	Suunnittelu	14
5.2	Toteutus	14
6	TOIMINNOT	15
6.1	Aloitussivu	15
6.2	Hakutoiminto	15
6.3	Kontin estimointi	16
6.4	Erilaiset näkymät	18
6.5	Repair -näkyvä	19
6.6	Tietokannan arkkitehtuuri	20
7	POHDINTA	21
8	JOHTOPÄÄTÖKSET	21
	LÄHTEET	23

1 JOHDANTO

Opinnäytetyön tarkoituksena oli suunnitella ja ohjelmoida logistiikka- sekä kuljetusyrityksen Bertschi Finland Oy:n jo olemassaolevaan web-sovellukseen lisäominaisuus. Työn tavoitteena oli muuttaa nykyinen paperillinen toimintatapa digitaaliseen muotoon. Web-sovellus kokonaisuudessaan käsittelee yrityksen kaikkia eri osa-alueita; kuljetuksia, tilauksia sekä asiakkaita, mutta tässä projektissa keskityttiin ainoastaan konttikorjaamoon.

Vaikka digitalisaatio on nykypäivää ja helpottaa palveluiden ja tuotteiden saantia sekä kehitystä, on silti edelleen toimialoja, jotka eivät ole täysin hyödyntäneet uusia teknologioita. Kuluttajille on jo muodostunut uusia odotuksia ja lähivuosina tullaan näkemään, kuinka yhä useampi perinteinen ala omaksuu digitalisaation hyödyt ja luo sen avulla uusia tuotteita ja palveluita. Myös liiketoiminta tulee kansainvälistymään tulevaisuudessa, joka entisestään on osasyynä uusien toimintatapojen suunnitteluun. (Digipore, 2020)

Opinnäytetyön päällimmäinen tarve oli logistiikkayrityksen työkalujen digitalisoiminen. Tarkoituksena oli siirtyä paperillisesta toiminnasta sähköiseen, paperittomaan muotoon, vähentää samalla paperin kulutusta sekä ylimääräistä työtä ja kustannuksia. Tämän ansiosta pystytään automatisoimaan toimenpiteitä. Työn prosessista haluttiin saada tehokkaampi ja vaivattomampi käyttäjälle sekä tallentaa dokumentit digitaaliseen muotoon. Sähköisessä muodossa olevaa tietoa on myös huomattavasti helpompi jakaa eteenpäin. Vanha prosessi vie itsessään runsaasti aikaa, työllistää ja kuluttaa paperia, koska kontin kaikki korjausvaiheet on kirjattava ylös paperille, jonka jälkeen ne on skannattava, kopioitava ja lähetettävä sähköpostilla. Opinnäytetyön web-sovelluksen automatisoitujen toimintojen ansiosta olisi mahdollista välttää helpommin virheitä, mutta haittana on sovelluksen omat toimintavirheet. Kustannuspuolella digitalisointi vähentää hallinnollista työtä.

Opinnäytetyön web-sovellus aloitettiin suunnittelemalla tarvittavat toiminnot ja sovelluksen perusrakenne. Web-sovellus on luotu Node.js -ympäristöön käyttäen hyödyksi Meteoria eli avoimen lähdekoodin ohjelmointialustaa, jota käytetään verkko-, mobiili- ja työpöytäkäyttöön (Meteor, 2020). Tässä projektissa käytettiin koodikielenä JavaScriptiä sekä Front-endin että Back-endin puolella. Sovelluksessa pystyttiin käyttämään samaa koodikieltä molemmissa, joka osoittautui huomattavaksi eduksi, koska pystyttiin esimerkiksi ajamaan tarkistus- ja laskentafunktiota sekä palvelimen että selaimen puolella. JavaScriptin lisäksi käytettiin apuna React-nimistä kirjastoa. Tietokantana käytettiin MongoDB:tä, joka on NoSQL-kanta (MongoDB, 2020).

Päädyin tähän opinnäytetyön aiheeseen, koska olin ollut jo aikaisemmin työharjoittelussa Bertschi Finland Oy:ssä, jossa oli suunniteltu konttikorjaamon digitalisoimishanketta. Opinnäytetyön tavoitteena oli tehdä web-sovelluksen testiversio sekä päätoiminnot, jotka ovat välttämättömiä kontin korjauksessa. Web-sovellusta aiotaan jatkossa kehittää lisää ja sitä on muokattava ja lisäominaisuuksia lisäävä, jotta saadaan mahdollisimman toimiva sovellus, joka vastaa työntilaajan tarpeita parhaimman mukaan. Web-sovellus on tarkoitus ottaa koekäyttöön lähiaikoina.

2 LYHENTEET JA MÄÄRITELMÄT

Web-sovellus	Internetin kautta käytettävä ohjelmisto.
Digitalisoituminen	Digitaalisen tietotekniikan yleistymistä arkielämän toiminnoissa.
Estimaatti	Esimerkiksi toiminnon tai esineen arvioimista.
ISO-standardit	Laadunhallinta järjestelmä.
Depot-palvelu	Tyhjien merikonttien varastointi-, nosto-, tarkistus- sekä korjauspalvelu.
Konttikorjaamo	Toimitila, eli tässä tapauksessa halli, jossa työskennellään konttien parissa.
Segmentti	Tarkoitetaan tässä tapauksessa kontin yhtä erillistä sivua.
Räjähdyskuva	Räjäytyskuva on kaavio, joka esittää esineen osat lievästi erotettuina, irrallaan toisistaan.
React	JavaScript-kirjasto käyttöliittymien rakentamiseksi.
JavaScript	Oliopohjainen ohjelmointikieli, jota nykyajan selaimet tukevat.
Node.js	Avoimen lähdekoodin JavaScript runtime-ympäristö JavaScript-koodin suorittamiseen palvelimella.
Meteor	Avoimen lähdekoodin alusta verkko, mobiili ja työpöytä käyttöön tukien JavaScriptiä.
MongoDB	Yleiskäyttöinen, asiakirjapohjainen ja hajautettu tietokanta, joka on rakennettu nykyaikaisille sovelluskehittäjille ja pilvipalveluille.
Front-end	Koodi, jota ajetaan verkkoselaimessa.
Back-end	Koodi, jota ajetaan palvelimen puolella sivuston tukena.
GitHub	Kehittäjille suunniteltu kehitysalusta versionhallintaa varten.
iTerm2	Vaihtoehto komentoriville.
SELF-JOIN	SQL-komento, joka on sidoksissa itseensä.

Metadata	Kuvailevaa ja määrittävää tietoa jostain tietovarannosta tai sisältyksiköstä.
Mean (MongoDB, Express.js, AngularJS, and Node.js)	JavaScript-pohjainen ohjelmointipakka dynaamisten web-sivujen ja ap- likaatioiden tekemiseen.

3 KÄYTETYT TEKNIIKAT

Web-sovelluksen kehittämisen helpottamiseksi käytettiin monia eri työkaluja. Kaikki työkalut olivat yritykselle entuudestaan tuttuja, sillä web-sovellusta oli kehitelty jo noin neljä vuotta. Tässä opinnäytetyössä olin mukana vain sovelluksen konttikorjaamo-osion kehittäessä.

3.1 Ohjelmointikieli

Web-sovelluksen ohjelmointikielenä käytettiin JavaScriptiä. JavaScript on pääasiassa web-ympäristössä toimiva Netscape Communications Corporationin kehittämä dynaaminen komentosarjakieli, joka mahdollistaa monimutkaisten ominaisuuksien toteuttamisen verkkosivuille (Wikipedia, 2020). JavaScript on tavanomaisten verkkosivutekniikoiden niin sanottu kolmas kerros, CSS:n ja HTML:n lisäksi (MDN contributors, 2020). JavaScriptin tärkein ominaisuus on dynaamisen toiminnallisuuden lisääminen web-sivuille ja se sallii asiakaspuolen skriptien interaktion käyttäjän kanssa, asynkronisen kommunikaation, selaimen rajoitetun hallinnan ja käyttäjälle näytettävän dokumenttisisällön muokkaamisen verkkoselaimessa (Wikipedia, 2020). JavaScriptin ohjelmointikieltä voidaan käyttää myös esimerkiksi palvelinten verkko-ohjelmoinnissa, tietokannoissa, kuten MongoDB ja CouchDB, sekä natiivi- ja mobiilisovellusten luomisessa (W3Schools, 2020). Koodia kirjoitettiin WebStorm-nimisessä sovelluksessa, joka on älykäs JavaScriptin ohjelmointiympäristö (JetBrains, 2020).

Web-sovelluksessa käytettiin tarkoituksella samaa ohjelmointikieltä sekä Front-endin että Back-endin puolella. Tämä helpotti saman koodin käyttöä selaimen sekä palvelimen puolella. JavaScript valikoitui palvelimen puolelle, koska Node.js ja Meteor käyttävät kyseistä ohjelmointikieltä. JavaScriptin käyttöä puolsi myös se, että suunnitellut tietokannat käyttivät samaa kieltä. Ohjelmointikielen apuna käytettiin myös JavaScriptille suunnattua kirjastoa, joka on tarkoitettu käyttöliittymän ohjelmointia varten. Sen avulla lataukset sekä muut käyttöliittymään liittyvät toiminnot hoituivat helpommin.

3.2 Tietokanta

Tietokanta on työkalu, jota käytetään tietojen keräämiseen ja järjestämiseen ja sinne voidaan tallentaa esimerkiksi henkilöihin, tuotteisiin ja tilauksiin liittyviä tietoja. Useat tietokannat ovat tekstinkäsittelyohjelmalla luotuja luetteloita tai laskentataulukoita ja luetteloiden kasvaessa tietoihin alkaa tulla päällekkäisyyksiä ja epäyhtenäisyyksiä. Tietoja voi olla vaikea ymmärtää luettelomuodossa ja tietojen osajoukoista voi olla hankalaa tehdä hakuja tarkistusta varten. Kun tällaisia ongelmia alkaa esiintyä, tiedot kannattaa siirtää tietokannan hallintajärjestelmällä (DBMS:llä), kuten Access, luotuun tietokantaan. (Microsoft, 2020)

Neljä vuotta sitten kun web-sovelluksen kehittäminen alkoi, harkittiin SQL:n käyttöä, mutta miettiessä sovelluksen toimivuutta ja skaalautuvuutta, päädyttiin NoSQL:n. SQL:n yhtenä haasteena olisi ollut SELF JOIN:it, eli data, joka alkaa toistamaan itseään ja menee puumaiseksi. Datan tahdottiin olevan puumaista, mutta sen hallittavuus ja ylläpidettävyyys SQL:ssä olisi mennyt liian monimutkaiseksi,

joten päädyttiin NoSQL-tietokantaan MongoDB-alustalla. NoSQL:llä voidaan myös tehdä poikkeuksia huomattavasti helpommin. MongoDB:stä voidaan siirtyä takaisin SQL:n pariin, mikäli on tarvetta. Datamallinnus olisi hyvä tehdä ennen kuin valitaan tiedokanta.

Lisäksi yhtenä syynä miksi kyseiseen tietokantaan päädyttiin oli se, että projekti aloitettiin MEAN-pakkaa käyttäen, jossa on MongoDB automaattisesti yhtenä keskiönä ja sen lisäksi MongoDB sekä Node.js toimivat hyvin yhdessä. Tämän vuoksi ei myöskään otettu post ressiä käyttöön. (Saksholm, 2020)

NoSQL-tietokannat eivät ole taulukkomuotoisia ja ne tallentavat tietoja eri tavalla kuin relaatiotaulut. NoSQL-tietokantoja on erityyppisiä niiden tietomallin perusteella. Tärkeimmät tyypit ovat asiakirja, avainarvo, leveä sarake sekä graafikanta. Ne tarjoavat joustavia kaavioita ja skaalautuvat helposti suurelle tietomäärälle ja suurelle käyttäjämäärälle. (Lauren Schaefer, 2020).

Yleinen väärinkäsitys on, etteivät NoSQL-tietokannat tai ei-relaatiotietokannat säilytä suhdetietoja hyvin. NoSQL-tietokannat voivat tallentaa suhdetietoja, mutta ne vain tallentavat sen eri tavalla kuin relaatiotietokannat. Monien mielestä SQL-tietokantoihin verrattuna NoSQL-tietokannoissa mallinnussuhteiden tiedot ovat helpompia, koska niihin liittyvää tietoa ei tarvitse jakaa taulukoihin. (Lauren Schaefer, 2020).

NoSQL- ja SQL-tietokantoja on vertailtu kaaviossa 1. (MongoDB, 2020)

MongoDB on asiakirjapohjainen hajautettu tietokanta, joka on kehitetty nykyaikaisille kehittäjille. Sen skaalattavuus ja joustavuus ovat avaintekijöitä ja niiden takia päädyttiin siihen. Käytettiin MongoDB:n apuna Studio 3T -sovellusta, joka on suunniteltu nimenomaan kyseiselle tietokannalle ja on myös ammattilaisille suunnattu graafinen käyttöliittymä, ohjelmointiympäristö sekä asiakasohjelma. (MongoDB, 2020)

	SQL Databases	NoSQL Databases
Tietojen tallennusmalli	Taulut, joissa on kiinteät rivit ja sarakkeet	Asiakirja:JSON-asiakirjat, Avainarvo:Avain-arvoparit, Laaja sarake: Taulut riveillä ja dynaamisilla sarakkeilla, Kaavio: Solmut ja reunat
Kehityshistoria	Kehitetty 1970-luvulla keskittyen tietojen päällekkäisyyden vähentämiseksi	Kehitetty 2000-luvun lopulla keskittyen skaalaamiseen ja mahdollistaen nopean sovellusmuutoksen sekä DevOps-käytäntöjen ohjaamisen
Esimerkit	Oracle, MySQL, Microsoft SQL Server, and PostgreSQL	Asiakirja: MongoDB ja CouchDB, Avainarvo: Redis ja DynamoDB, Leveä sarake: Cassandra ja HBase, Kuvio: Neo4j ja Amazon Neptune
Päätarkoitus	Yleiskäyttöinen	Asiakirja:yleiskäyttöinen, Avainarvo: Suuret tietomäärät yksinkertaisilla hakukyselyillä, Laaja-sarake: Suuret tietomäärät ennustettavilla kyselyillä, Kaavio: yhdistettyjen tietojen välisten suhteiden analysointi ja yhdistäminen
Malli	Kiinteä	Joustava
Skaalautuvuus	Pystysuuntainen	Vaakasuuntainen
ACID-tapahtumat	Tuetut	Useimmat eivät tue ACID-tapahtumia. Jotkut kuten MongoDB - kuitenkin tukevat
Joins	Tyypillisesti vaadittavat	Tyypillisesti ei vaadita
Tiedot objektikartoitukseen	Vaatii ORM (objektikohtainen kartoitus)	Monet eivät vaadi ORM: ää. MongoDB-asiakirjat kartoittavat suoraan tietorakenteita suosituimmissa ohjelmointikielissä.

Kaavio 1, SQL- ja NoSQL-tietokantojen vertailu.

3.3 Palvelin

Projekti toimii Node.js ympäristössä käyttäen samalla Meteor-nimistä alustaa. Tätä valintaa tuki se, että molemmat ovat JavaScript-pohjaisia ja niistä oli jo aikaisempaa kokemusta.

Koko ohjelma pyörii erillisellä pilvipalvelimella, jota ohjattiin iTerm2-nimisen sovelluksen kautta. iTerm2 on perinteisen komentorivi apuohjelman korvaaja, koska se tuo kaikki uudet ja modernin ominaisuudet käyttöön, mikä ei perinteisellä komentorivillä onnistu. Tärkeä seikka on myös se, että iTerm2 on suunniteltu toimimaan ainoastaan macOS-laitteilla. (iTerm2, 2020)

3.4 Versionhallinta

Projektin suuruuden ja sen parissa työskentelevien työntekijöiden takia oli myös tärkeää pitää huolta versionhallinnasta ja siitä huolehtii GitHub. GitHubia voi käyttää verkon kautta tai natiivin työpöytäsovelluksen kautta. Sen avulla on helppo pysyä ajan tasalla sekä tehdä erilaisia huomautuksia tai tehtävänantoja projektin parissa työskenteleville. Tämän avulla vältettiin projektin rikkoutuminen ja voitiin hallita projektin kulkua. Esimerkiksi virhetilanteessa voitiin aina palata takaisin edelliseen toimivaan versioon. (GitHub, 2020)

4 ASIAKAS

4.1 Työn tilaaja

Opinnäytetyön tilaajana oli sveitsiläisen perheyriityksen Bertschi AG:n omistama tytäryhtiö Suomessa, Bertschi Finland Oy, joka on perustettu vuonna 1956. Bertschi Finland Oy:n toimitusjohtaja on Janne Tiusanen. Bertschi AG on Euroopan johtava kemian teollisuuden, logistiikan ja intermodaaleihin bulkki kuljetuksiin erikoistunut yhtiö ja se tarjoaa kuljetus-, varastointi- ja lisäarvopalveluita petrokemianteollisuudelle. Bertschi Finland Oy avasi rautatielogistiikkaterminaalin Vuosaareen satamaan vuonna 2011 ja se työllistää Vuosaaren terminaalissa noin 35 henkilöä ja on painottunut raakamuovin käsittelyyn, varastointiin sekä säiliökonttien väliaikaiseen varastointiin. Terminaalin kalustoon kuuluu kaksi konttikurattajaa, kuusi rekkaa, terminaalitraktori sekä erilaisia trukkinostimia. (Tiusanen, 2020)

4.2 Konttikorjaamo

Konttikorjaamot tarjoavat tyhjen merikonttien varastointi-, nosto-, tarkistus- ja korjauspalveluita. Merikonttien on täytettävä ISO-standardit, joten konttien kunto on tarkistettava säännöllisesti sekä ylläpidettävä rekisteriä tarkistetuista yksiköistä. Depot-palvelut ovat suuri kuluerä ja liiketoiminnan kasvu on suoraan verrannollinen liikkuvaan konttimäärään. Kontit eivät tuota liikevaihtoa kun ne varastoidaan tyhjinä ja lisäksi suuret korjaukset seisottavat yksiköitä. (Tiusanen, 2020)

Aiemmin Bertschi Finland Oy osti korjaamotoiminnan muualta palveluna, jolloin kontit oli kuljetettava palveluntarjoajan tiloihin. Tämä synnytti paljon ylimääräisiä kustannuksia sekä toimintavarmuuden heikkenemistä, joten vuonna 2015 yritys investoi omaan korjaamohalliin, jolloin tyhjen konttien tarkistukset, korjaukset sekä sisäpussien asennustyöt palveluntarjoajan toimesta voitiin tehdä omissa tiloissa. Tällä pyrittiin poistamaan tyhjen yksiköiden kuljettamisesta syntyneet kustannukset sekä vähentämään tyhjen konttien varastointikustannuksia. Vuonna 2017 sopimus palveluntarjoajan kanssa kuitenkin purettiin ja korjaamotoiminta kotiutettiin, koska konttikorjaamon toiminta ei toiminut odotetulla tavalla eikä palveluntarjoaja ei pystynyt tarjoamaan palveluitaan halutulla tasolla. Tämän jälkeen korjaamolle palkattiin esimies sekä kaksi korjaajaa. (Tiusanen, 2020)



Kuva 1. Konttikorjaamohalli.

Konttihuollossa on **viisi** päävaihetta:

1. Korjaamopäällikön vastuulla on **suunnittelu, kapasiteetin ja resurssien suunnittelu**, esimerkiksi on mietittävä, kuinka monta konttia on pihalla ja monta korjaajaa tarvitaan. Tärkeää on siis operatiivisen toiminnan suunnittelu eli miten korjaamo pystyy tuottamaan kontteja tarpeeksi tai mistä korjaamolle saadaan tarpeeksi kontteja.
2. **Konttien tarkastaminen** eli mitä korjataan, jos korjataan.
3. **Konttien korjaaminen** ja korjausten dokumentointi.
4. **Konttien pussittaminen**; kontti pussitetaan aina kun se menee korjaamolle, mutta kontti voidaan pussittaa myös ilman korjausta.
Kun pussi poistetaan, vanha pussi hävitetään ja kierrätetään. Uuden pussin asennuksen on aina onnistuttava ennen kontin lastausta. Kontti on myös puhdistettava ja tarkastettava tiivisteet. Pussit ovat kertakäyttöisiä taatakseen mahdollisimman hyvän laadun ja niiden tarkoituksena on suojata tuotteita.
5. **Dokumentointi ja laskuttaminen.** Konttikorjaamon perinteisen paperillisen toimintatavan korvaamisen taustalla on digitalisoituminen, eli paperillisesta toiminnasta siirtyminen sähköiseen muotoon, jonka ansiosta pystytään automatisoimaan toimenpiteitä. Kun tieto on sähköisessä muodossa, on sitä helpompi levittää ja jakaa eteenpäin. Digitalisointi vähentää hallinnollista työtä ja siten kustannuksia. Paperillinen toimintatapa ei ole kustannustehokasta, sillä siinä toimintaan kuuluu papereiden täyttöö, kopiointia, skannausta ja lähettämistä, jolloin se vie enemmän työaikaa ja on hidasta. Manuaalisen tavan olisi tarkoitus jäädä pois digitalisoinnin myötä.

5 WEB-SOVELLUS

Web-sovellus koostuu kahdesta pääpainikkeesta, joista päästään korjaamon aloitussivulle sekä Repair-sivusta, johon on listattu eri statuksilla olevia korjauksia. Korjaamotoiminto sisältää useamman eri sivun sekä vaiheen. Seuraavissa luvuissa käsitellään kokonaisuuden suunnittelua sekä toteutusta. Toiminnot -luvussa päästään tarkastelemaan konttikorjaamon web-sovelluksen toimintoja tarkemmin (katso luku 6).

5.1 Suunnittelu

Sovelluksen lähtökohtana oli olla helppokäyttöinen ja sisältää mahdollisimman vähän eri työvaiheita. Tarkoituksena olisi automatisoida kaikki toiminnot, mitkä vain olisivat mahdollisia.

5.2 Toteutus

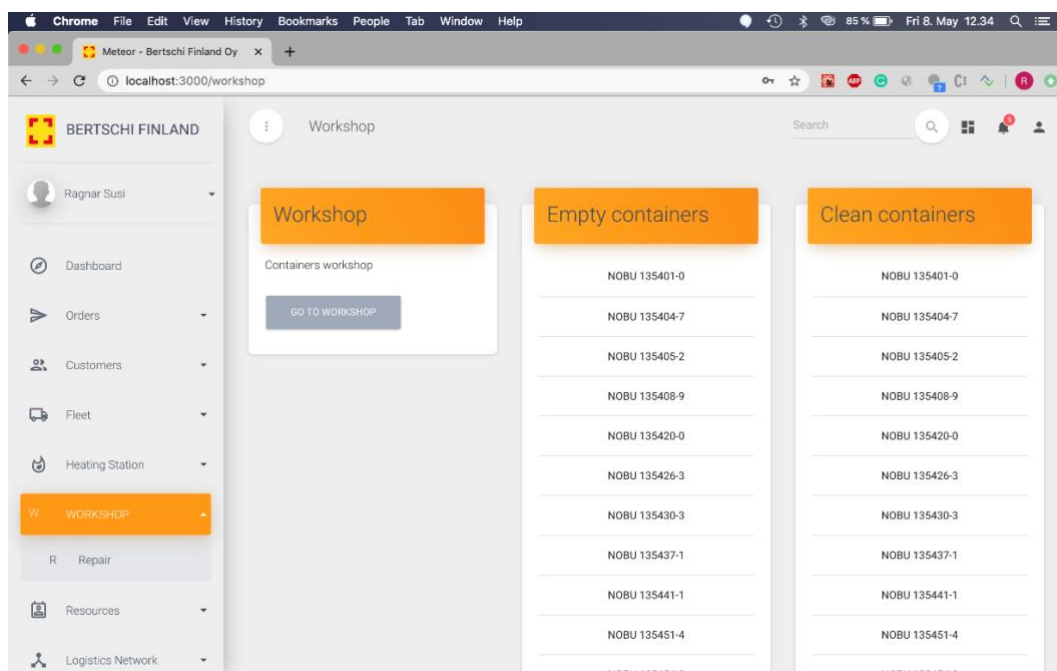
Web-sovellusta lähdettiin toteuttamaan tutustumalla ensimmäisenä sovelluksessa käytettäviin työkaluihin. Projekti oli ollut käynnissä jo noin neljä vuotta, joten yleisimmät työkalut olivat jo tiedossa. Kun oli saatu päätös käytettävistä työkaluista, päästiin aloittamaan itse käyttöliittymän tekeminen. Ensimmäisenä tehtiin aloitussivu ja siihen painike, josta pystyy siirtymään itse korjaamotyökalun käyttöön. Aloitussivun jälkeen tehtiin muut tärkeät sivut ja lähdettiin hahmottelemaan ulkoasua. Kun ulkoasu oli saatu halutun näköiseksi, pystyttiin tekemään sivuille erilaisia toimintoja sekä Front-endin että Back-endin puolelle.

6 TOIMINNOT

Konttikorjaamon nykyiset huoltotoimenpiteet vaativat monia eri vaiheita sekä paljon ylimääräistä työtä, jonka takia suunniteltiin ja kehitettiin web-sovelluksesta mahdollisimman itsenäinen, jotta käyttäjä ei joutuisi tekemään ylimääräisiä toimenpiteitä ja sovellus osaisi ennakoida sen, mitä käyttäjä haluaa. Automatisoinnin avulla voidaan täyttää vakioarvoja automaattisesti, käsinkirjauksen sijaan. Kaikki sovelluksen toiminnot, näkymät sekä listat ovat vasta väliaikaisia ratkaisuja ja niitä tullaan muuttamaan käyttäjän tarpeen mukaan. Sovellus on tehty sen suunnitelman pohjalta mikä olisi paras ja hyödyllisin käyttäjälle.

6.1 Aloitusivu

Aloitussivu on nimensä mukaisesti ensimmäinen näkymä, josta voidaan aloittaa kontin huoltotoimenpiteet sovelluksen avulla. Sivulla on painike josta päästään valitsemaan estimoitava kontti (Kuva 2). Näkymään on myös lisätty listoja, joista voidaan tarkastella eri statuksilla olevia kontteja. Esimerkiksi kuvan mukaisesti tyhjiä sekä puhtaita kontteja. Tulevaisuudessa tullaan lisäämään listoihin myös niitä kontteja, joita halutaan jatkuvasti tarkastella.



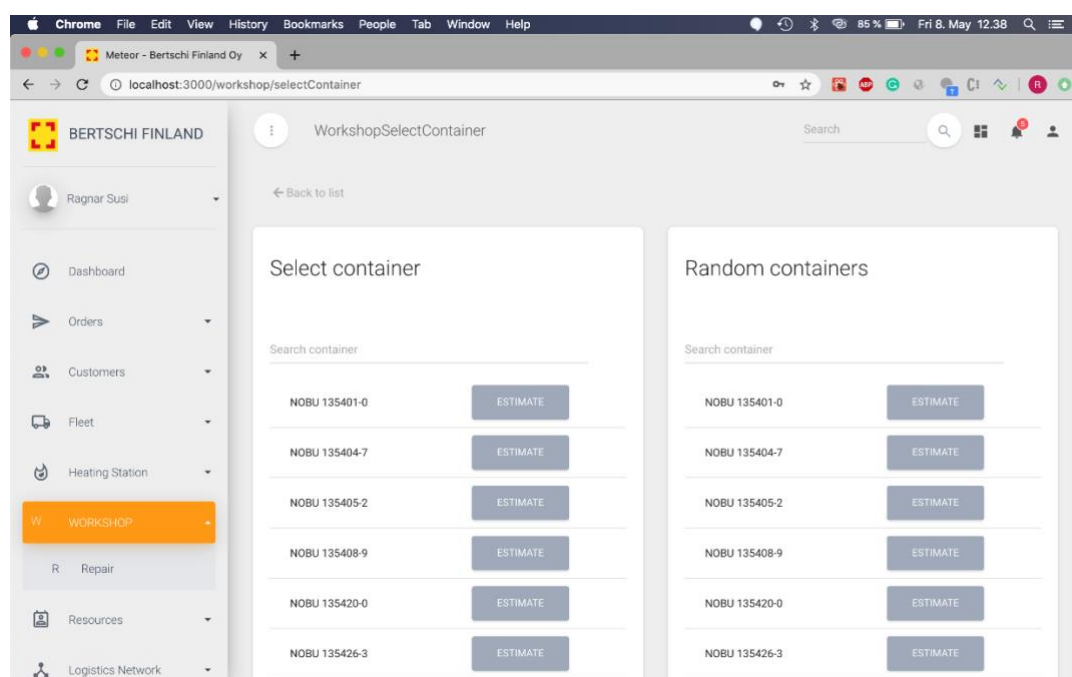
Kuva 2. Toiminnon aloitusivu.

6.2 Hakutoiminto

Hakutoimintoa voidaan käyttää estimoitavan kontin hakuun. Mahdolliset korjaukseen tulevat kontit ovat listattuna, mutta koska kontteja on niin paljon, tehtiin myös hakutoiminto haun helpottamiseksi. Kontti voidaan valita joko listalta tai hakea älykkäällä haulla, jolloin hakukenttään voidaan kirjoittaa mikä vain kohta kontin numerosta, joka sen avulla näyttää kaikki yhteensopivat vaihtoehdot. Listaus

osaa itse suodattaa pois sellaiset kontit, jotka eivät ole tulossa korjaukseen. Kun kontista ruvetaan tekemään estimaattia, kontti häviää listalta.

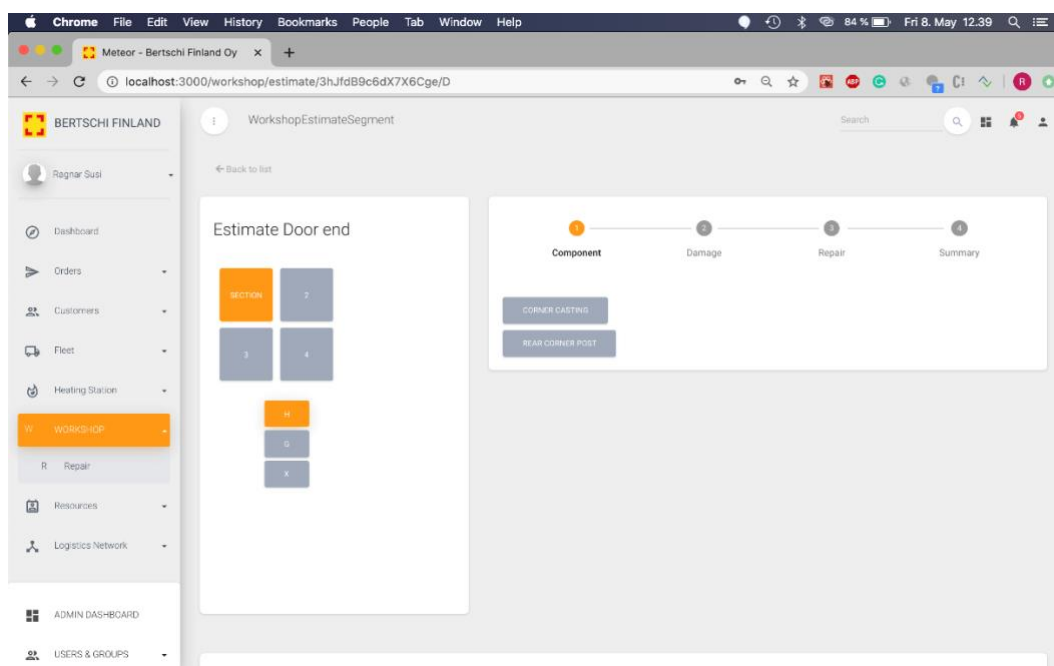
Hakutoiminnon huonona puolena ovat kirjoitusvirheet. Huoltotoimenpiteen tekemisessä on syytä olla tarkkana ja yritys itsessään on jo painottanut tarkkaavaisuuteen ja huolellisuuteen. Konttinumeron syöttämisessä voi aina käydä virhe ja sen minimoimiseksi tehtiin haku, joka karsii kontteja aina syötettyjen numeroiden tai kirjaimien mukaan. Esimerkiksi jos syöttää hakukenttään kontin numeron ”NOBU 1355”, se ehdottaa kaikkia samanalkuisia kontteja.



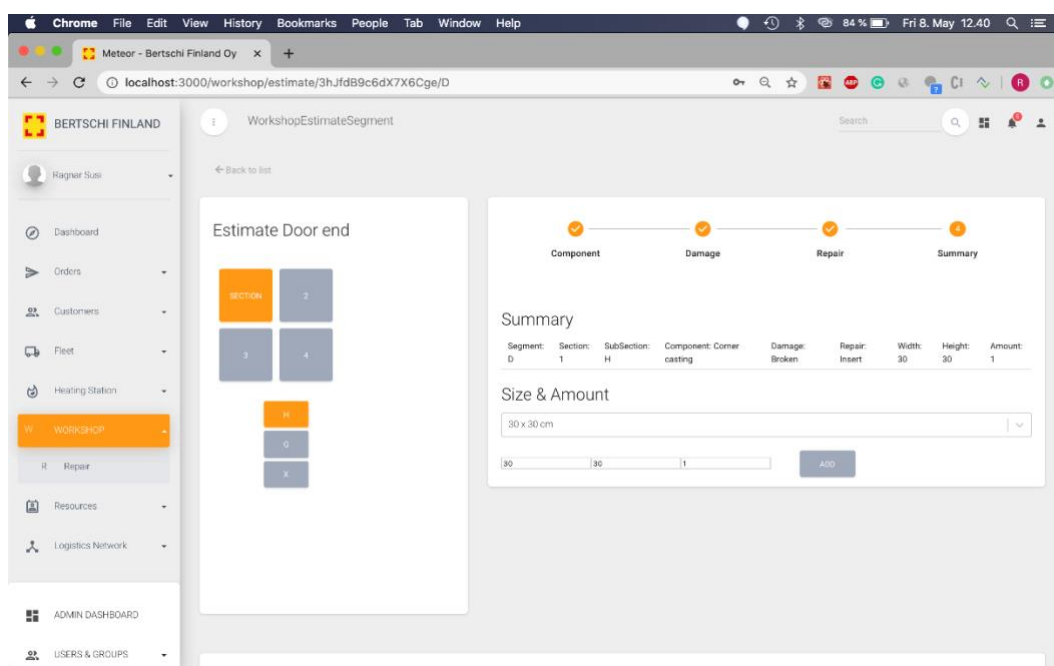
Kuva 3. Näkymä, josta päästään valitsemaan työstettävä kontti.

6.3 Kontin estimointi

Yhtenä suurimpana toimenpiteenä on kontin estimointi. Valittuaan kontin avautuu käyttäjälle räjähdyskuva kontista, joka on kaavio, mikä esittää kontin osat toisistaan erotettuina ja irrallaan toisistaan. (Kuva 4). Se sisältää kontin jokaisen sivun erikseen. Valittua kontin sivun, aukeaa näkymä, jossa kyseinen sivu on vielä tarkempuna räjähdyskuvana. Kun kontin segmentti on estimoitu muuttaa se väriä (Kuva 5). Kyljet, katto, lattia ja pohja ovat jaoteltu kuuteen eri segmenttiin ja päädyt neljään eri segmenttiin. Segmenteistä voidaan valita haluttu alue, johon korjaus tulisi tehdä, jonka jälkeen valitaan vielä tarkempi sijainti, vika, korjaustoiminta ja kuluva korjausmateriaalin määrä. Tämä kaikki tallennetaan yhtenä estimaattirivinä, josta tehdään korjaus ja jota käytetään koko prosessin ajan eri näkymissä.



Kuva 6. Näkymä, josta valitaan vaurion sijainti, komponentti, vaurio, korjausmenetelmä sekä siihen kuluvan materiaalin määrä.

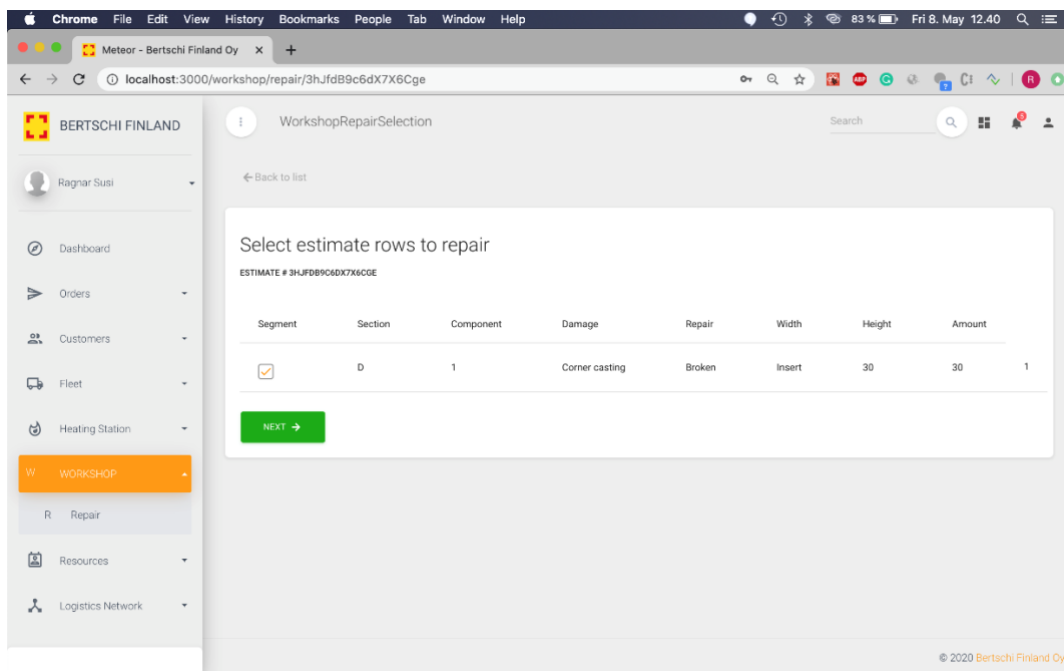


Kuva 7. Kontin estimointiprosessin viimeinen vaihe eli valitaan korjaukseen menevän materiaalin määrä sekä yhteenveto estimaatista.

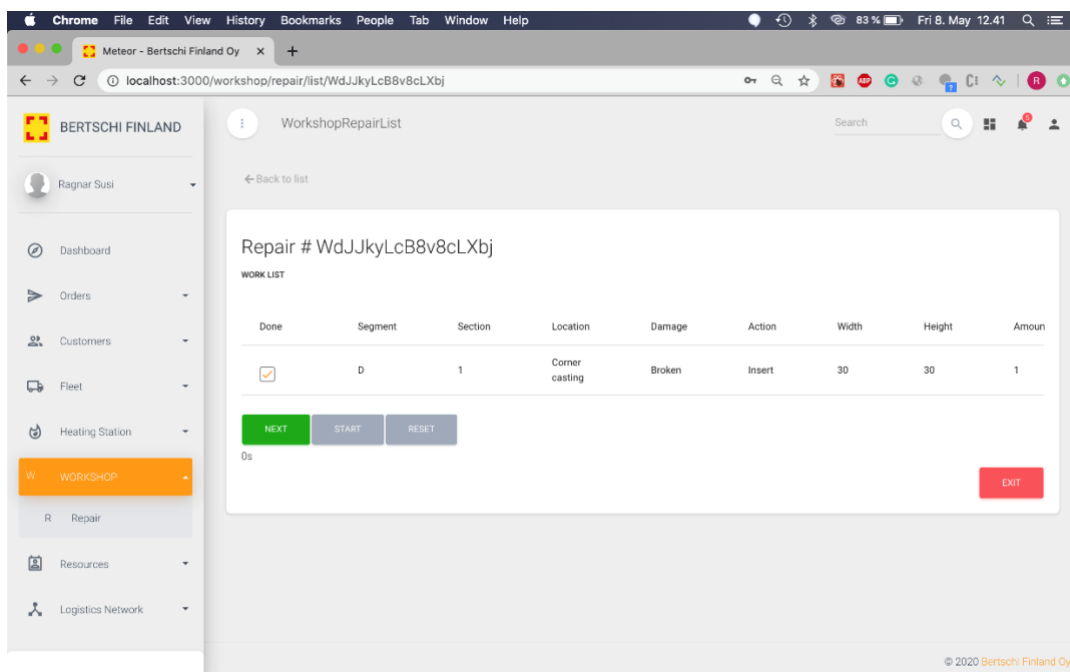
6.4 Erilaiset näkymät

Estimaattitoiminnon jälkeen on tehty kolme yksinkertaisempaa näkymää, joissa näkyy estimaatista tehdyn korjauksen tiedot ja joita voidaan tarvittaessa muokata. Ensimmäinen näkymä on konttikorjaamon esimiehelle, joka valitsee korjattavat rivit korjaajalle. Tämän jälkeen se siirtyy korjaajalle, joka näkee mitä, mistä ja miten pitää korjata. Korjaaja voi itse muokata myös tietoja.

Viimeisenä vaiheena on tarkastusvaihe, jossa esimies tarkistaa lopullisen työn tiedot ja hyväksyy ne, ja jonka jälkeen lähtee suoraan asiakkaalle lasku ja dokumentit omaan arkistoon.



Kuva 8. Valitaan estimoidun kontin korjattavat vauriot.

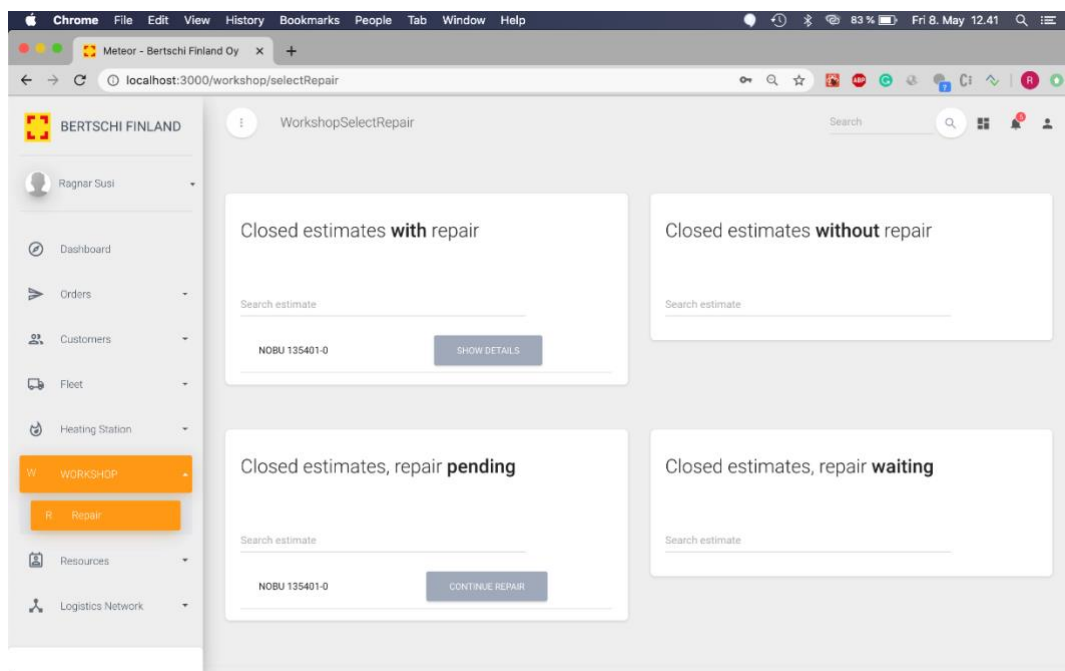


Kuva 9. Korjaajan näkymä, josta näkyy korjaustoimenpiteen tiedot.

6.5 Repair -näkymä

Repair-sivun näkymän on tarkoitus näyttää erilaisia listoja konttien estimaateista (Kuva 10). Sivulla näkyy listattuna esimerkiksi estimaatteja, joista on tehty jo korjaukset ja sellaisia, joista korjauksia ei ole vielä tehty. Tämä on tehty sitä varten, että voidaan katsoa tehtyjen estimaattien tietoja tai

jatkaa keskeneräisten tekemistä. Tämä mahdollistaa estimaatin tai korjauksen lopettamisen missä vaiheessa toimenpidettä vain.



Kuva 10. Repair -näkyvä, jossa voi tutkia tehtyjä korjauksia tai jatkaa keskeneräisiä.

6.6 Tietokannan arkkitehtuuri

Tietokantaobjektin suunnittelu oli myös tarkkaa ja aikaa vievää. Tiedon pitää olla tietokannassa halutussa muodossa ja sen pitää sisältää kaikki halutut arvot. Sovelluksessa päädyttiin NoSQL tietokantaan sen puumaisuuden ja käytettävyyden takia. Tietokanta toimii MongoDB alustalla, joka tallentaa tiedot JSON muotoisena objektina (Kuva 11). Tämä oli tehokkain tapa työskennellä datan kanssa, sillä se tukee suuria määriä dataa ja sisäkkäisiä objekteja arvioina (MongoDB, 2020).



Kuva 11. JSON muodossa oleva objekti (MongoDB, 2020).

7 POHDINTA

Web-sovelluksen ei suunniteltu valmistuvan täysin opinnäytetyön teon aikana, sillä projekti on hyvin laaja ja ominaisuuksia on vielä paljon lisättävänä. Projekti onnistui lähes kaikin puolin suunnitelmien mukaan. Sovelluksen testiversio saatiin melkein valmiiksi, mutta sen käyttöä oikeassa toimintatilanteessa ei vielä ehditty tehdä.

Oppimisen kannalta olisi ollut mielenkiintoista saada palautetta sovelluksen toimivuudesta sekä puutteista. Web-sovelluksen kehittäminen jatkuu vielä, mutta opinnäytetyön ulkopuolella.

Työn alku oli hidas, koska liityin jo käynnissä olevaan projektiin, joka hankaloitti omalta osaltaan työn aloitusta. Sovelluksen teossa oli paljon uutta opeteltavaa, jota varten jouduin hakemaan ja lukemaan tietoa niin, että pystyin työskentelemään itsenäisesti projektin parissa.

Jatkoa ajatellen olisin voinut tehdä käytännön kokeiluja enemmän, mikä olisi varmasti ollut opettavaisempaa. Omasta mielestäni omalla alallani kehittyä parhaiten tekemällä, mutta itse kulutin alussa aikaa tiedon hankintaan enemmän kuin koodaamiseen.

Projektissa meni hienosti Front-end suunnittelu ja toteutus. Opin mielestäni kaikki pakolliset toiminnot ja ohjelmat pystyväkseni etenemään projektissa. Mitä enemmän pääsin tekemään ja harjoittelemaan käytännössä erilaisia toimintoja, sitä nopeammin opin. Projektin alussa pystyin hyödyntämään koulussa oppimaani tietoa.

Projektissa oli jonkin verran haasteita ja varsinkin Back-endin puolella. Palvelimen puolella koodaaminen oli mielestäni haastavampaa, koska siellä on paljon erilaisia tarkistusfunktioita ja muita toimintoja, jotka on otettava tietoturvasyistä huomioon.

8 JOHTOPÄÄTÖKSET

Opinnäytetyön tavoitteena oli luoda JavaScript pohjainen web-sovellus konttikorjaamolle. Sovelluksen on tarkoitus korvata nykyinen paperillinen toimintatapa, joten sovelluksen täytyy toimia mahdollisimman virheettömästi ja sujuvasti.

Lopputuloksena saatiin tehtyä web-sovellus, joka piti sisällään kaikki suunnitellut pääominaisuudet ja toiminnot toimivat halutulla tavalla. Koekäyttöön sovellus ei vielä kuitenkaan päätenyt, mutta on muutamia pieniä korjauksia vaille valmis testattavaksi. Tavoitteena olisi lähiaikoina saada web-sovellus koekäyttöön, jotta siitä saadun palautteen perusteella voitaisiin aloittaa sovelluksen jatkokehitys, käyttäjien tarpeiden mukaiseksi. Ennen tätä tulee kuitenkin saada hyväksyttävä päätös konttikorjaamon esimieheltä, joka käy läpi sovelluksen ja tarkistaa, että sieltä löytyy kaikki tarvittavat ominaisuudet konttien korjauksien toimenpiteitä varten.

Kehitettävää ja paranneltavaa on kuitenkin vielä paljon tiedossa tämän web-sovelluksen suhteen.

LÄHTEET

Digipore, 2020. Digitalisaation tulevaisuus. Viitattu: 10.04.2020. - <https://kokokansandigi.fi/digitalisaation-tulevaisuus/>

Meteor, 2020. FAQ. Viitattu: 10.04.2020. - <https://www.meteor.com/meteor-faq>

MongoDB, 2020. NoSQL Explained. Viitattu: 10.04.2020. - <https://www.mongodb.com/scale/nosql-explained>

Wikipedia, 2020. JavaScript. Viitattu: 15.04.2020. - <https://fi.wikipedia.org/wiki/JavaScript>

MDN contributors, 2020. A high-level definition. Viitattu: 20.04.2020. - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

W3Schools, 2020. Why Study JavaScript? Viitattu: 21.04.2020. - <https://www.w3schools.com/js/>

Microsoft, 2020. Mikä on tietokanta? Viitattu: 24.04.2020. - https://support.office.com/fi-fi/article/perustiedot-tietokannasta-a849ac16-07c7-4a31-9948-3c8c94a7c204#__toc257378454

Saksholm, Joni, 2020. Bertschi Finland Oy. Viitattu: 14.05.2020.

Lauren Schaefer, 2020. What is NoSQL? Viitattu: 14.05.2020. - <https://www.mongodb.com/nosql-explained>

MongoDB, 2020. NoSQL vs SQL Databases. Viitattu: 14.04.2020. - <https://www.mongodb.com/nosql-explained/nosql-vs-sql>

iTerm2, 2020. What is iTerm2? Viitattu: 20.04.2020. - <https://www.iterm2.com/index.html>

GitHub, 2020. GitHub. Viitattu: 02.04.2020. - <https://github.com/>

Tiusanen, Janne, 2020. Bertschi Finland Oy Manager.

MongoDB, 2020. Etusivu. Viitattu: 01.04.2020. - <https://www.mongodb.com/>

JetBrains, 2020. Etusivu. Viitattu 26.05.2020. - <https://www.jetbrains.com/webstorm/>