Markus Saronsalo

Vierailijahallintajärjestelmä yritysvieraille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

9.5.2023

Tiivistelmä

Tekijä:	Markus Saronsalo
Otsikko:	Vierailijahallintajärjestelmä yritysvieraille
Sivumäärä:	32 sivua + 2 liitettä
Aika:	9.5.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mobile Solutions
Ohjaajat:	Tietohallintopäällikkö Taia Gorski-Kokko Yliopettaja Hannu Markkanen

---

Insinöörityön tavoitteena oli kehittää vierailijahallintajärjestelmä elintarvikealan yritykselle JavaScriptin React-kirjastoa käyttäen. Vierailijahallinnassa käytettävän ohjelmiston käyttötarkoituksena oli kerätä yrityksessä vierailevien henkilöiden henkilötietoja ja tallentaa saapumis- ja poistumisajankohta. Ohjelmisto toteutettiin verkkoselaimessa toimivalla käyttöliittymällä ja sen kanssa yhdessä toimivan Node.js-ympäristössä toimivan palvelimen kanssa.

Tietosuojalain noudattaminen on olennainen osa henkilötietojen tallennusta, säilyttämistä ja käsittelyä. Insinöörityön toteutus kehitettiin mukailemaan lain vaatimuksia mahdollisimman tarkasti. Toteutuksessa luotiin isäntäyrityksen lähiverkossa toimiva suojattu tiedonsiirto HTTPS-protokollaa ja fetch API -rajapintaa käyttäen. HTTPS:n käyttöön luotiin juurivarmenne ja SSL-sertifikaatit. Henkilötiedot tallennettiin JSON-tiedostoissa palvelimena ja käyttäjärajapintaa ajavan tietokoneen paikalliseen tallennustilaan Ohjelmisto toimi käyttökokemuksen osalta sulavasti, ja insinöörityön toteutuksen osalta tavoitteet saavutettiin tyydyttävästi.

Tietosuojaan liittyvät vaatimukset eivät saavuttaneet haluttua tasoa insinöörityön toteutuksen kehittämiseen varatun aikataulun puitteissa. Ohjelmiston ominaisuudet tietojen tallennuksen ja käsittelyn osalta todettiin vajavaisiksi tietosuojalain osalta. Ohjelmisto ei valmistunut tuotantokäyttöön sopivaksi, mutta siitä huolimatta lopputuloksena kuitenkin valmistui toimiva kokonaisuus prototyyppiasteella.

Insinöörityön kehittämisen aikana saaduista tuloksista voidaan päätellä, että Reactilla on mahdollista riittävillä resursseilla toteuttaa kohtuullisen helposti henkilötietojen keräämiseen soveltuva ohjelmisto.

Avainsanat: JavaScript, React, ReactJS, rajapinta, päätelaite, kirjasto, tietoturva

Abstract

Author:	Markus Saronsalo
Title:	Visitor management system for company visitors
Number of Pages:	32 pages + 2 appendices
Date:	9 May 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Mobile Solutions
Supervisors:	Hannu Markkanen, Principal Lecturer Taia Gorski-Kokko, IT Manager

The goal of this study was to develop a visitor management system for a food company operating in the food industry, using a JavaScript library React. The intended use case of the program used for visitor management was to collect and save personal information from company visitors and to log their arrival and departure times. The software was implemented as a browser-based frontend interface working in together with a server running in the Node.js runtime environment.

Obeying the data protection law is an essential part of saving and handling personal information. The implementation of the software was developed to comply with the law as precisely as possible. In the implementation, a secure connection working within the parent company's LAN was created, using the HTTPS-protocol and the fetch API interface. A root certificate and SSL certificates were created for the use of HTTPS. The personal information was saved in JSON-files to the local drive of the computer used for running the server and the user interface. The software worked in terms of user experience and the goals regarding the thesis' implementation were achieved satisfactory.

The requirements related to data protection did not reach the desired level within the schedule reserved for the development of the software. The features of the software were found to be lacking regarding the data protection law. The software was not finished to fit for production use, but nevertheless, a functional entity was completed at a prototype level.

Based the results obtained during the development of the software, it can be concluded that with React it is possible with sufficient resources to implement software suitable for collecting personal data in a reasonably easy manner.

Keywords: JavaScript, React, ReactJS, interface, terminal, library, information security

Sisällys

Lyhenteet ja käsitteet

CA: *Certificate Authority*. Luotettavia TLS/SSL-varmenteita jakava taho.

CSS: *Cascading Style Sheets.* Internet sivujen rakenteen ja tyylittelyn määrittelyyn käytetty koodikieli.

GDPR: *General Data Protection Regulation*. Henkilötietojen käsittelyä säätelevä laki, jota alettiin soveltaa kaikissa EU-maissa keväällä 2018.

HTML: *HyperText Markup Language*. Avoimesti standardoitu merkintäkieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä eli hypertekstiä.

HTTP: *Hypertext Transfer Protocol.* Siirtoprotokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.

HTTPS: *Hypertext Transfer Protocol Secure.* Käytetään salattuun tiedonsiirtoon. Samankaltainen kuin HTTP-protokolla, mutta käyttää SSL/TLS-salausta.

IDE: *Integrated Development Environment*. Ohjelmointiympäristö eli ohjelma tai joukko ohjelmia, joiden avulla suunnitellaan ja toteutetaan ohjelmistoa.

SSL Secure Socket Layer. Salausprotokolla tietoliikenteen suojaamiseen.

TLS *Transport Layer Security*. Salausprotokolla tietoliikenteen suojaamiseen. Secure Socket Layer -protokollan seuraaja.

VSC: *Visual Studio Code.* Microsoftin kehittämä avoimen lähdekoodin koodieditori.

1 Johdanto

Insinöörityössä kehitettiin JavaScriptin kirjastolla Reactilla vierailijahallintajärjestelmä toimeksiantajalle Ab Chipsters Food Oy:lle, jonka toimipiste sijaitsee Keravalla, Jäspilässä. Yritys toimii elintarvikealalla erilaisten kulutushyödykkeiden valmistajana ja jälleenmyyjänä.

Yrityksen tiloissa, joista isossa osassa vaaditaan korkeaa hygieniaa, vierailee useasti ulkopuolisia henkilöitä ja heihin liittyviä tietoja kerätään talteen, jotta voidaan varmistaa turvallinen elintarvikehygienia. Vierailijoiden sisäänkirjautumisessa kerätään talteen muun muassa vierailun ajankohta, henkilötiedot ja mahdolliset terveydelliset seikat, jotka voivat vaikuttaa vierailun kestoon ja laatuun.

Insinöörityössä toteutetun ohjelmiston tavoitteena oli saada toimintaan toimeksiantajalle mahdollisimman käyttäjäystävällinen ja tietoturvallinen web-päätteellä toimiva ohjelmisto, jota käytetään yrityksessä vierailevien henkilöiden terveydentilaan liittyvien tietojen ja henkilötietojen keräämisessä ja säilyttämisessä. Ohjelmiston tärkeimpinä vaatimuksina olivat sen mahdollisimman helppokäyttöinen käyttöliittymä ja tietoturvallisuus. Näiden vaatimusten täyttämiseen käytettiin apuna pääasiallisesti Reactille saatavia kirjastoja ja HTTPS-protokollaa käyttävää suojattua tiedonsiirtoa.

Raportissa perehdytään ensin yleisesti henkilötietojen keräämiseen liittyviin tietosuojalain määritelmiin (luku 2). Raportin seuraavassa osassa kerrotaan tarkemmin ohjelmistoon liittyvistä vaatimuksista ja sen suunnittelusta (luku 3). Seuraavassa luvussa käydään läpi ohjelmistossa käytettyjä teknologioita, jotka ovat pääasiallisesti Reactin kanssa käytettyjä kirjastoja (luku 4). Luvussa 5 käydään läpi insinöörityön toteutukseen liittyviä asioita, joita ovat muun muassa eri ohjelmiston osat ja niiden toiminnallisuus (luku 5). Lopuksi analysoidaan insinöörityön toteutuksessa saavutettuja tuloksia ja tehdään yhteenveto (luku 6).

2 Tietosuojalaki

Tässä luvussa käydään läpi tietosuojaperiaatteita ja säädöksiä, joita jokaisen henkilötietoja käsittelevän ihmisen tai organisaation täytyy tietää. Rikkomukset näitä säädöksiä vastaan voivat johtaa jopa rangaistustoimenpiteisiin tietosuojalain 26. artiklan [1] mukaan.

2.1 Yleistä asiaa henkilötiedoista

Henkilötietoja ovat kaikki tiedot, jotka liittyvät tunnistettuun ja tunnistettavissa olevaan henkilöön. Näitä tietoja ovat esimerkiksi nimi, puhelinnumero ja sijaintitiedot. [2.]

Rekisterinpitäjä on ihminen tai organisaatio, joka määrittelee, mihin tarkoitukseen ja millä tavalla henkilötietoja käsitellään. Rekisterinpitäjäksi voidaan määritellä esimerkiksi potilastietoja käsittelevä yritys, sairaala tai verkkokauppa. [2.] Rekisteriä insinöörityössä kehitetyn ohjelmiston tallentamista henkilötiedoista pitää työn toimeksiantaja Ab Chipsters Food Oy.

Henkilötiedoilla voi myös olla käsittelijä, joka on ihminen tai organisaatio. Käsittelijä käsittelee henkilötietoja rekisterinpitäjän puolesta. Käsittelijä voi olla esimerkiksi toisen yrityksen markkinointia hoitava markkinointitoimisto tai IT-palveluntarjoaja, jolla on pääsy rekisterinpitäjän henkilötietoihin. [2.] Insinöörityön toteutuksessa kerättyjen tietojen käsittelyssä ei käytetä ulkopuolista tahoa, vaan kaikki henkilötietojen käsittely on yrityksen henkilökunnasta nimettyjen vastuuhenkilöiden vastuulla.

Tietosuojaperiaatteen mukaisia henkilötietoja on käsiteltävä lainmukaisesti, asianmukaisesti ja rekisteröidyn kannalta läpinäkyvästi. Tietoja tulee käsitellä luottamuksellisesti, turvallisesti ja vain tarpeellisissa määrin henkilötietojen käsittelyn tarkoitukseen nähden. Niitä on kerättävä ja käsiteltävä tiettyä, nimenomaista ja laillista tarkoitusta varten. Epätarkat ja virheelliset henkilötiedot on poistettava

tai oikaistava viipymättä ja päivitettävä aina tarvittaessa. Tiedot täytyy säilyttää ainoastaan niin kauan, kuin on tarpeen. [2.]

2.2 Henkilötietojen keräämisen merkitys

Insinöörityössä kehitettävän vierailijahallintajärjestelmän asianmukaisen toiminnallisuuden edellytys on, että tietosuojalain alaisia henkilötietoja kerätään ja tallennetaan. On ensiarvoisen tärkeää, että tietosuojalakia noudatetaan tarkasti, jotta voidaan estää henkilötietojen vuotaminen väärien henkilöiden tietoon. Henkilötietojen kerääminen ja käsittely helppokäyttöisellä ja nopeasti toimivalla päätelaitteella sujuvoittaa yritysvieraiden vierailuja ja mahdollistaa niiden luotettavan kirjaamisen. Vierailijoiden luovuttamien henkilötietojen lisäksi myös saapumis- ja poistumisajat kirjataan järjestelmään. Poikkeustilanteissa, kuten tulipalon sattuessa, voidaan selvittää yritykseen kuulumattomien henkilöiden mahdollinen senhetkinen oleskelu yrityksen toimitiloissa ja varmistaa, että he ovat poistuneet rakennuksesta turvallisesti.

Ab Chipsters Food Oy:ssä elintarvikkeita käsittelevänä laitoksena on laaja kokonaisuus korkeahygieenistä aluetta, ja siitä syystä on ensiarvoisen tärkeää minimoida mahdollisuus esimerkiksi tartuntatautien leviämiseen tai rajata toimitilojen ja elintarvikkeiden saastumisen mahdollinen alkulähde. Vierailunhallintajärjestelmänä toimivan ohjelmiston yksi tehtävistä on kerätä tietoja yritysvieraiden terveydentilasta siltä osin, että sen voidaan todeta olevan hyväksyttävällä tasolla vierailua varten eikä käyntiä estävää mahdollisesti tarttuvaa tautia, joka voisi estää pääsyn korkeahygieeniselle alueelle, ole tiedossa tai havaittavissa. Todenmukaisen ja toimivan tiedonkeruun edellytyksenä kuitenkin on, että yritysvierailija ilmoittaa rehellisesti oman arvionsa terveydentilastaan. Mahdollisissa epäselvissä tapauksissa vierailu voidaan yrityksen käytännön mukaan kieltää kokonaan tai rajata vain niihin yrityksen tiloihin, joissa vierailijan terveydentila ei aiheuta terveydellisiä riskejä muille henkilöille.

Vaikka terveystietoja kerätään, on kuitenkin mahdollista, että talon ulkopuolinen henkilö tietämättään levittää esimerkiksi tartuntatautia. Järjestelmän avulla

kerättyjen henkilötietojen avulla voidaan jälkeenpäin selvittää, kuka yrityksessä on vieraillut minäkin ajankohtana. Jokaisella vierailijalla on oletusarvoisesti yrityksen henkilökunnassa henkilö, jota hän on saapunut tapaamaan. Tämä henkilö määritellään vierailijan kirjautuessa järjestelmään. Hän huolehtii sekä vieraan ohjauksesta että asianmukaisesta sisään- ja uloskirjautumisesta. Jos yrityksen toimitiloista on poistuttava vaaratilanteen sattuessa, täytyy vierailusta vastaavan henkilön huolehtia asianmukaisesta uloskirjautumisesta.

Tietosuojaperiaatteet ilmoitetaan ohjelmiston käyttäjälle ennen henkilötietojen luovuttamista ohjelmiston käyttöön erikseen määritetyllä tietosuojalausekkeella, joka henkilötietojaan yritykselle luovuttavan henkilön täytyy hyväksyä, jotta tietoja on luvallista tallentaa yrityksen hallitsemaan rekisteriin. Tietosuojalausekkeessa ilmoitetaan muun muassa tietoja keräävän yrityksen nimi, tiedonkeruun tarkoitus ja tietojen säilyttämisen kesto.

Insinöörityön ohjelmiston avulla kerätyt henkilötiedot säilytetään oletusarvoisesti yhden vuoden ajan, eikä niitä muokata ilman asianomaisen henkilön pyyntöä. Tiedot säilytetään yrityksen hallittavana ja poistetaan ilman erillistä pyyntöä. Henkilötiedoista voidaan kuitenkin tarpeen mukaan selvittää vierailuihin liittyviä tietoja. Tietoja säilytetään turvallisessa sijainnissa, johon ei ole pääsyä asiaankuulumattomilla henkilöillä.

3 Ohjelmiston vaatimukset ja suunnittelu

Vierailijahallintajärjestelmän suunnitteluvaiheessa käytiin läpi toimeksiantajayrityksessä toimivan ohjaajan ja IT-asiantuntijan kanssa ohjelmistoon liittyviä tarpeita ja käytettävissä olevien resurssien määrää. Suunnittelun pohjalta voitiin todeta, että insinöörityössä pystytään toteuttamaan vähintään prototyyppi, jota ei tulla käyttämään talon ulkopuolisten henkilöiden kirjaamiseen, mikäli kaikki tietoturvallisuuteen liittyvät vaatimukset eivät täyty. Prototyyppiä apuna käyttäen selvitettäisiin toimivuus ja kehitettäisiin ratkaisuja siihen, millä tavalla, missä muodossa ja mihin sijaintiin vierailijoiden henkilötietoja voidaan tallentaa käytännöllisesti, niin että tietosuojalakia noudatetaan tarkasti.

Ohjelmiston rakenne suunniteltiin alustavasti, ja asiakaspäätteen käyttöliittymän osalta päädyttiin alustavasti seuraavaan rakenteeseen:

- etusivu
- tietosuojalomake
- ohjesivu
- lisäohjeistus
- lomake.

Käyttäjä navigoi ohjelman sisällä sivu kerrallaan, ja hänelle jaetaan vierailuun liittyvää ohjeistusta jokaisella sivulla. Etusivulla käyttäjälle esitetään toimeksiantajayrityksen ja samoissa tiloissa toimivan Baron's Food -yrityksen logot. Etusivulta käyttäjä ohjataan tietosuojalomakkeeseen, jonka hyväksyminen on edellytys jatkamiseen ohjesivulle, jossa käyttäjälle jaetaan yleistä tietoa yrityksen vierailuun liittyvistä käytännöistä. Lisäohjeistussivulla vierailijalle jaetaan tarvittaessa lisätietoa sen perusteella, missä yrityksen tiloissa henkilö tulee käyntinsä tarkoituksen ja terveystilanteensa perusteella vierailemaan. Lopulta hän täyttää henkilötietonsa lomakkeeseen Lomake-sivulla ja luovuttaa tiedot yrityksen haltuun.

Ohjelmistoon kehitettiin myös erillinen hallintapaneeli, joka mahdollistaa tallennettujen henkilötietojen tutkimisen. Sen käyttö kuitenkin rajattiin käyttäjätunnuksella, ja sen verkkopolku ei ole nähtävissä ohjelmistoa käyttäville vierailijoille.

Työn edistyessä kartoitettiin, mitkä ovat valmiin ohjelmiston pakollisia ja valinnaisia tarpeita. Ohjelmistoon liittyviä ominaisuuksia päätettiin lisätä ja kehittää tarpeen mukaan sen perusteella, mitkä olisivat mahdollisia toteuttaa insinöörityön aikataulun ja käytettävien resurssien puitteissa. Mikäli kaikkia kehitystavoitteita ei saavutettaisi insinöörityön toteutuksen aikana, ohjelmiston kehittämistä jatkettaisiin uudella aikataululla siihen pisteeseen, että se voitaisiin lopulta ottaa yritysvierailijoiden käyttöön.

Suunnitteluvaiheessa haasteeksi ilmeni se, että yritykseltä puuttui oma palvelinympäristö eikä varsinaista ohjelmiston päätelaitteeksi sopivaa tietokonetta

todennäköisesti saataisi käytettäväksi insinöörityön tekemisen aikana. Tästä syystä palvelimena toimiva ohjelmisto päätettiin kehittää itse. Tämä ohjelmisto olisi käyttöliittymästä erillään toimiva toteutus, joka olisi yhteydessä vierailijoiden käyttämän rajapinnan kanssa ja hoitaisi henkilötietoihin liittyvien tiedostojen vastaanottamisen ja käsittelemisen. Käyttäjärajapintana toimivan sopivan päätelaitteen puuttumiseen mietittäisiin ratkaisua työn edetessä, ja se liitettäisiin osaksi ohjelmistoa, mikäli aikataulu sallisi.

Toimeksiantajayrityksellä oli käytettävissä tilapäisiksi päätelaitteiksi soveltuvia tietokoneita, joiden avulla ohjelmiston käyttökokemuksen ja toimivuuden testaus voitaisiin suorittaa. Palvelimen ja käyttöliittymän ohjelmisto suoritettaisiin yrityksen työkäytöstä poistetulla Windows 10 -pohjaisella pöytätietokoneella.

Henkilötietojen turvallinen siirtäminen päätelaitteelta palvelimelle on olennainen osa ohjelmistoa tiedonkeruun lisäksi. HTTP-protokolla valittiin alustavasti tiedonsiirtoon, mutta se ei suojaamattomuutensa takia soveltunut muuhun kuin testikäyttöön. HTTPS-protokolla valikoitui ratkaisuksi tähän ongelmaan. HTTP:llä ja HTTPS:llä toimivat toteutukset ovat toisiinsa verrattuna samankaltaisia Reactissa, joten kun henkilötietojen siirtäminen päätelaitteelta palvelimelle saatiin toimintaan HTTP-protokollan avulla, muokattiin ohjelmisto sen jälkeen käyttämään HTTPS-protokollaa. Lisäturvaksi tiedonsiirtoon kaikki tietoliikenne päätelaitteen ja palvelimen välillä rajattiin vain yrityksen omaan lähiverkkoon, johon asiaankuulumattomilla henkilöillä ei ole pääsyä.

Henkilötietojen tallentamiseen mietittiin kolmea vaihtoehtoa: tietojen tallentamista tietokantaan tietueina, JSON-tiedostoina verkkopalvelimelle tai palvelinkoneen kiintolevylle. Tietokannan toteuttaminen oli varteenotettava vaihtoehto, mutta henkilötietojen tallentaminen JSON-tiedostoihin valikoitui helppoutensa takia lopulliseksi tallennustavaksi. JSON-tiedostosta nähdään esimerkki liitteessä 1. Tiedostojen lopullinen rakenne muotoutui mahdollisimman yksinkertaiseksi, ja ne sisältävät henkilötiedot helposti luettavassa muodossa. Palvelinkoneen paikallinen tallennustila valikoitui ensisijaiseksi henkilötietojen

tallennussijanniksi vierailijahallintajärjestelmän prototyypin toteutuksessa. Verkkopalvelimen käyttöönottoa mietittäisiin, jos aikataulu sen sallisi.

4 Käytetyt ohjelmistot ja teknologiat

Vierailunhallintajärjestelmän toteuttamiseen käytetyksi ohjelmointikieleksi valikoitui React, joka on Facebookin kehittämä JavaScriptin avoimen lähdekoodin kirjasto. React käyttää ns. HTML-in-JavaScript-syntaksia (JSX) [3]. Tässä luvussa kerrotaan ohjelmistossa käytetyistä tekstieditoreista ja muutamista olennaisista kirjastoista.

4.1 Ohjelmointiin käytetyt koodieditorit

Insinöörityössä käytetyt koodieditorit olivat Visual Studio Code (VSC) ja Brackets. VSC on monipuolinen editori, joka tukee ohjelmistokehityksen toimintoja, kuten virheenjäljitystä, tehtävien suorittamista ja versionhallintaa [4]. VSC:tä käytettiin React-projektin alustamiseen ja ohjelmiston rakenteen luomiseen kotiympäristössä ennen sen siirtämistä GitHubiin ja sieltä yrityksen käyttöympäristöön.

Brackets asennettiin ja otettiin käyttöön palvelin- ja asiakasrajapintaa ajavalle koneelle yrityksen toimitiloissa. Brackets on avoimen lähdekoodin ilmainen tekstieditori, joka on kevyt, mutta sisältää paljon ominaisuuksia [5]. Se tarvitsee VSC:hen verrattuna huomattavasti vähemmän keskusmuistia Windows 10:n tehtävienhallinnasta löytyvien prosessitietojen perusteella. Tätä tietoa ei kuitenkaan vahvistettu muuten kuin vertaamalla muistin käyttöä kahdella eri tietokoneella, joten vertailutulokset voivat vaihdella eri tilanteissa. VSC asennettiin kotiympäristössä toimivalle tietokoneelle. Bracketsia käytettiin yrityksen omassa kehitysympäristössä. VSC:n muistinkäyttö oli insinöörityötä koodattaessa vähintään 500 Mt, kun vastaavassa tilanteessa Brackets varasi sitä käyttöönsä vähimmillään alle 20 Mt. Kun Bracketsia käytettiin yrityksen kehitysympäristössä, ohjelmistoa pystyi kehittämään samalla, kun se oli toiminnassa. Tämä menettely säästi aikaa poistamalla tarpeen siirtää jokainen pienikin päivitys GitHubin

kautta käyttöön ohjelmistossa. Merkittävät päivitykset kuitenkin vietiin GitHubiin Git Bash -terminaalia käyttäen.

4.2 Käyttäjärajapinnan kirjastot

Yhtenä käyttöliittymän kirjastoista toimi Material UI (MUI). Se on avoimen lähdekoodin CSS:n (Cascading Style Sheets) tapainen komponenttikirjasto Reactille, jota voidaan käyttää mukauttamaan ohjelmiston käyttöliittymän rakennetta ja ulkonäköä. Se perustuu Googlen Material Design -suunnittelumalliin. MUI sisältää valmiita React-komponentteja, kuten painikkeita, valikkoja ja kenttiä. [6.] MUI:n avulla toteutettiin suurin osa insinöörityön tyylittelystä. Esimerkkikoodissa 1 näytetään, miten MUI:n avulla voidaan määritellä luokalle tyylittelyä.

```
const useStyles = makeStyles((theme) => ({
     buttonContainer: {
     display: “flex”,
     justifyContent: “space-between”,
     marginTop: “20px”,
});
```

Esimerkkikoodi 1. Ote insinöörityössä käytetystä tyylittelystä buttonContainer-luokalle MUI:n makeStyles-funktiota käyttäen.

Esimerkkikoodissa 1 esitetään yksinkertainen muotoilupohja, joka määrittelee sen sisälle asetettavien elementtien sijoittelun. Tässä tapauksessa pääasialliset käyttökohteet ovat painikkeet. Tyylimäärittelyn sisälle asetetut elementit jakautuvat käytettävissä olevaan tilaan tasaisesti rinnakkain. Esimerkiksi kahta painiketta käytettäessä ne asettuvat tasapuolisesti käyttöliittymässä ruudussa vasemmalle ja oikealle justifyContent-ominaisuuden määrittelyn ”space-between” mukaan.

Esimerkkikoodissa 1 mainittu tyylimäärittely voidaan ottaa käyttöön ohjelmassa käyttämällä esimerkkikoodissa 2 esitettyä tapaa, joka on ote sivulta Ohjesivu.

```
import useStyles from "../styles";

const classes = useStyles();

<div className={classes.buttonContainer}>
```

Esimerkkikoodi 2. Tyylit tuodaan käyttöön sivulle import-lausekkeella. React-komponentin sisällä määritellään classes-muuttuja, joka luodaan MUI:n makeStyles-funktiosta.

Esimerkkikoodissa 2 Import-lauseke tuo useStyles-funktion tiedostosta styles, joka sijaitsee suhteellisessa polussa yhtä tasoa ylemmässä kansiossa. Funktiota kutsutaan classes-muuttujan kautta. Käyttöön tuotu tyylimäärittely buttonContainer otetaan käyttöön div-elementissä määrittelemällä sen className-attribuutti täsmääväksi haluttuun tyyliluokkaan.

Navigointi ohjelmiston eri sivujen välillä toteutettiin React Routerin avulla. Perinteisillä internetsivuilla selain pyytää dokumentin web-palvelimelta, lataa ja määrittelee siihen liittyvät CSS- ja JavaScript-resurssit ja näyttää selaimelle palvelimen lähettämän HTML-tiedoston. Tämä toistuu aina uudestaan jokaisen uudelleenladatun sivun kohdalla. React Router on kehys, jonka avulla ohjelma voi päivittää näkyvissä olevan sivun osia ilman, että tarvitaan pyyntö kokonaan uudelle dokumentille, ja se pystyy myös käyttämään hyödykseen selaimen historiaa. [7.]

4.3 Palvelinpuolen kirjastot

Express on palvelinpuolelle tarkoitettu Node.js-pohjainen kehityskehys. Se tarjoaa vankan valikoiman ominaisuuksia web- ja mobiilikehitykseen [8]. Expressiä käytettiin insinöörityön toteutuksessa palvelimella fetch API:n ja käyttöliittymän välisen tietoliikenteen toteuttamiseen. Express alustettiin jäsentämään (parse) siihen sisäänrakennetulla väliohjelmalla (middleware) pelkästään JSON-tiedostoja määrittämällä app.use(express.json(().

Multeria käytettiin JSON-tiedostojen vastaanottamisessa. Se on Node.js-väliohjelma multipart/form-data-mediatyypin käsittelyyn [9]. Insinöörityön toteutuksessa henkilötietolomakkeen tiedot lähetettiin palvelimelle multipart/form-data-

muodossa. Lomakkeella lähetettyjen tekstikenttien arvot tuotiin käytettäväksi palvelimelle tulevista body- ja file-objekteista. Lomakkeelle syötetyt tiedot sisältyvät body-objektiin, ja file-objekti sisältää viittauksen lähetettyyn tiedostoon. Näitä objekteja käytettiin palvelimella fetch API:n lähettämien tietojen purkamisessa ja käsittelyssä.

5 Vierailijahallintajärjestelmän toteutus

5.1 Käyttäjärajapinta ja palvelin

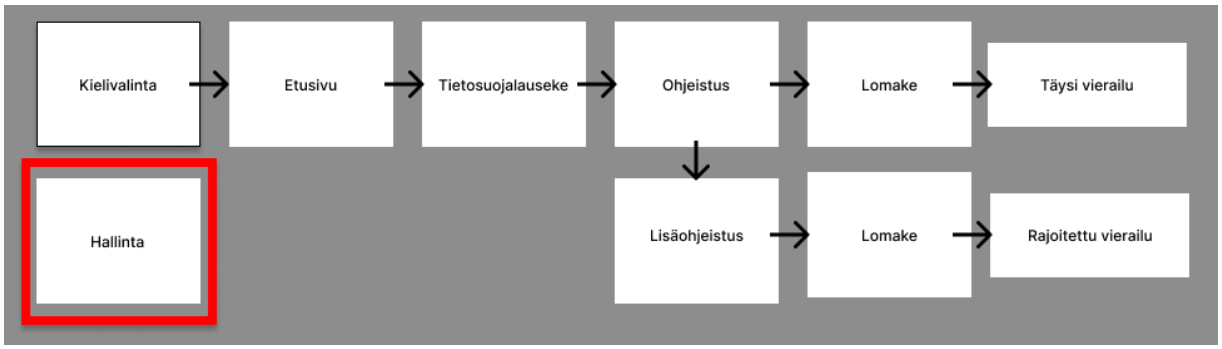
Vierailijahallintajärjestelmään luotiin kaksi erillistä osaa: käyttäjärajapinta Visitorhandler ja palvelinpuoli VisitorhandlerServer. Tämä erottelu tehtiin, jotta palvelimen logiikka ja käyttäjärajapinta voitiin toteuttaa erillisinä projekteina. Käyttäjärajapinta toimii palvelinohjelmistoa huomattavasti laajempana kokonaisuutena, joka sisältää suurimman osan insinöörityön logiikasta ja sisällöstä. Visitorhandler kattaa muun muassa ohjelmiston käyttämät sivut, funktiot, teemat ja komponentit. VisitorhandleServer kattaa henkilötietojen käsittelemiseen ja tallentamiseen liittyvän logiikan.

Ohjelmiston koodaamisessa käytettiin apuna TeamViewer-etähallintaohjelmaa. Se valittiin käyttöön luotettavuuden ja tehokkuuden takia. Se toimii eri laitteiden etäkäyttöön tarkoitettuna ohjelmana, joka on käytössä sadoilla miljoonilla käyttäjillä [10]. Etäkäyttö helpotti ohjelmiston kehittämistä, koska se poisti tarpeen olla aina paikan päällä yrityksen tiloissa, kun ohjelmistoa koodattiin.

5.2 Navigaatio ja monikielisyys

Vierailijahallintajärjestelmän käyttökokemuksen helppouden takaamiseksi navigointi pyrittiin toteuttamaan rakenteellisesti lineaarisesti. Tämä ilmenee ohjelmistossa siten, että käyttäjä käy yksi kerrallaan läpi tietosuojalausekkeen, vierailuun liittyvän ohjeistuksen ja lomakkeen. Yleistä ohjeistusta jaetaan käyttäjälle jokaisella sivulla. Ohjelmistoa käyttäessään vierailija voi parhaimmillaan navigoida ensimmäiseltä sivulta viimeiselle sivulle poikkeamatta kertaakaan

määritellyltä reitiltä. Ohjelmiston navigaatio visualisoidaan kuvassa 1 nähtävällä kaaviolla.



Kuva 1. Vierailijahallintajärjestelmän navigointikaavio. Nuolet esittävät etenemistä järjestelmässä. Hallintapaneeli on vain määritellyn rekisterinpitäjän saavutettavissa.

Kuvan 1 kaavio havainnollistaa ohjelmiston eri osia laatikoina. Nuolet esittävät käyttäjän liikkumista ohjelmiston sisällä. Pääsy hallintapaneeliin on kielletty muilta kuin erikseen määritetyiltä henkilöiltä. Tämä havainnollistetaan paksulla punaisella reunalla Hallinta-laatikon ympärillä. Kun käyttäjä aloittaa ohjelmiston käytön, hänelle esitetään ensin sivu Kielivalinta, jossa voidaan valita ohjelmiston käyttämä kieli. Se valitaan suomen, englannin ja ruotsin väliltä. Kielivalinnan jälkeen käyttäjä ohjataan sivulle Etusivu, jossa esitetään Ab Chipsters Food Oy:n tiloissa toimivien yrityksien logot. Tällä sivulla käyttäjä voi painaa Sisään-nappia, joka avaa tietosuojalausekkeen modal-elementtiin. Käyttäjä voi joko hyväksyä lausekkeen ehdot tai kieltäytyä niistä. Mikäli käyttäjä kieltäytyy ehdoista, henkilötietoja ei voida kerätä. Tässä tilanteessa käytännön mukaan vierailua yrityksessä ei voida järjestää.

Jos käyttäjä hyväksyy tietosuojalausekkeen, hänet ohjataan sivulle Ohjesivu. Tällä sivulla käyttäjälle esitetään yleistä ohjeistusta yrityksessä vierailemista varten. Ohjesivulla käyttäjältä tiedustellaan nykyinen terveydentila ja vaaditaan vakuutus, että hän on omasta mielestään terve eikä koe muutenkaan olevansa estynyt vierailemaan. Mikäli estettä vierailulle ei ole, käyttäjä voi siirtyä eteenpäin Lomake-sivulle. Kun käyttäjä on täyttänyt ja lähettänyt lomakkeen, hänen

tietonsa tallennetaan yrityksen käytettäväksi ja hän voi siirtyä yrityksen tiloihin vastuuhenkilönsä ohjaamana.

Mikäli kirjautumisen aikana ilmenee, että vierailua ei voida turvallisesti suorittaa, käyttäjä ohjataan Lisäohjeistus-sivulle. Käyttäjää ohjeistetaan ottamaan yhteyttä henkilöön, jonka kanssa hän on sopinut tapaamisen. Vierailija voi tässä tapauksessa yrityksen henkilökunnan harkinnan mukaan päästä vierailemaan rajoitetusti yrityksen tiloissa.

Ab Chipsters Food Oy:n henkilökunnassa ja asiakkaina on henkilöitä, jotka puhuvat eri kieliä. Tästä syystä ohjelmistoon lisättiin valittavaksi käytettävä kieli kolmesta vaihtoehdosta: suomi, englanti ja ruotsi. Eri kielet määriteltiin luomalla locales.js-tiedosto, johon määriteltiin eri kielillä tekstimuuttujia (string), jotka näytetään käyttäjän valitseman kielen mukaan.

Esimerkkikoodissa 3 näytetään yksi tapa määrittää eri teksteille muuttujia kolmella eri kielellä locales-tiedostoon.

```
***locales.js
const locales = {
    fi: {
tervetuloa_otsikko: "Tervetuloa Ab Chipsters Food Oy:n
tuotantolaitokseen. Toivomme, että viihdytte luonamme"
},
    en: {
tervetuloa_otsikko: "Welcome to Ab Chipsters Food Oy's
production facility. We hope you enjoy your visit."
},
    sv: {
tervetuloa_otsikko: "Välkommen till Ab Chipsters Food Oy:s produktion-
sanläggning. Vi hoppas att du trivs under ditt besök."
     }
}

export default locales;
```

Esimerkkikoodi 3. Määritellään kolmelle eri kielelle string-muuttuja nimeltä tervetuloa_otsikko, joka vaihtuu valitun kielen perusteella.

Esimerkkikoodissa 3 nähdään kolmelle kielelle muuttujat tallennettuna muuttujan locales sisälle ja määritellään tässä tapauksessa etusivulla nähtävä string-muuttuja tervetuloa_otsikko.

Eri kielivalintojen esittämiseen valittiin React Intl -kirjasto. Kirjaston käyttämä IntlProvider-komponentti otettiin käyttöön App.js-tiedostossa, jotta käännökset olivat käytettävissä koko sovelluksessa. Komponenttia FormattedMessage apuna käyttäen esitetään erikseen määritelty tekstimuuttuja tiedostossa locales.js olevien käännösten perusteella. Esimerkkikoodissa 4 esitetään React Intl -kirjaston käyttö App.js-tiedostossa.

```
***App.js
import { IntlProvider } from "react-intl";
import { FormattedMessage } from "react-intl";
import messages from "./locales.js";

const [locale, setLocale] = useState("fi"); // oletuskieli
<IntlProvider locale={locale} messages={message{locale}]>
<FormattedMessage id="tervetuloa_sisään" defaultMEssage="Sisään"/>
```

Esimerkkikoodi 4. React Intl -kirjaston ”IntlProvider”- ja ”FormattedMessage”-komponenttien käyttöönotto ja niiden määrittely toimintaan ohjelmistossa.

Esimerkkikoodissa 4 tuodaan import-lausekkeella moduulit IntlProvider ja FormattedMessage käyttöön App.js-tiedostossa. Lisäksi tuodaan messages-objekti, joka sisältää eri kielillä käytettävät käännökset. Määritetään tila (state) locale ja alustetaan se tekstillä ”fi”, joka asettaa locales.js-tiedostosta suomen ohjelmiston oletuskieleksi.

Sen jälkeen IntlProvider-komponentille jaetaan kieliasetus propseina (properties). Propsit ovat ominaisuuksia, jotka toimivat tiedonvälittäjinä Reactissa eri komponenttien välillä. Tämä tapahtuu määrittämällä locale={locale}, joka on alustettuna suomi eli ”fi”. Eri kielet kattavat käännökset jaetaan käyttäen tuotua messages-objektia, joka sisältää käännökset eri kielille. Esimerkkikoodissa 4 nähdään myös FormattedMessage-moduulin käyttö. Tässä tapauksessa käytetään locales.js:n tervetuloa_sisään-tietokentän string-muuttujaa. Tapauksessa, jossa tietokenttää vastaavaa muuttujaa ei ole tiedostossa saatavilla, defaultMessage-parametri määrittelee näkyvän viestin.

React Router vaati toimiakseen propseja myös RouteConfig-komponenttiin, jotta locale-ominaisuus ja kielivalinnan vaihtava changeLocale-ominaisuus

saatiin käyttöön myös ohjelmiston muissa tiedostoissa. Ohjelmistossa käytettiin senhetkisen kellonajan ja päivämäärän hakemista käyttämällä setInterval-funktiota 1000 ms:n välein ja konstruktoria new Date. Tällä tavalla haettu tieto tallennettiin dateTime-tilaan, joka piti myös jakaa RouteConfig-komponentille. Tämä mahdollisti tietokoneen oman tarkan kellonajan ja päivämäärän käyttämistä ohjelmistossa. Esimerkkikoodissa 5 nähdään ote RouteConfig-komponentin määrittelystä.

```
***App.js
<RouteConfig dateTime={dateTime} changeLocale={changeLocale} lo-
cale={locale} />
```

Esimerkkikoodi 5. Jaetaan dateTime-, changeLocale- ja locale-ominaisuudet React Routerin lapsikomponentille RouteConfig.

Esimerkkikoodissa 5 nähdään koodi App.js-tiedostosta, joka jakaa propsit dateTime, changeLocale ja locale lapsikomponentille RouteConfig. Tällä määrittelyllä ne saatiin toimintaan koko ohjelmiston laajuudella.

Ohjelmiston kielen valinta tehtiin ensimmäisellä sivulla. Kuvassa 2 nähdään ote sivun Kielivalinta näkymästä, jossa valitaan ohjelmiston käytössä oleva kieli.

Kuva 2. Kielivalintasivu, jossa voidaan valita ohjelmiston käyttämä kieli. Vaihtoehtoina ovat suomi, englanti ja ruotsi.

Kuvan 2 Suomen, Englannin ja Ruotsin liput kuvastavat ohjelmiston kielivalintoja. Kuvassa havainnollistetaan käyttäjän valitsema kieli englanti ja tämä

valinta esitetään lipun tyylittelyn suuremmalla kirkkaudella (brightness). Lippujen alla olevasta napista ”CHOICE OK” valittu kieli voidaan ottaa käyttöön ohjelmistossa. Käyttäjän on myös mahdollista vaihtaa kieli jälkeenpäin käyttöliittymään toteutetusta sivun yläreunassa olevasta valikosta.

5.3 Henkilötietojen lähetys palvelimelle

Palvelimen tehtävänä oli kuunnella porttia 443 ja ottaa hyväksytystä lähteestä vastaan kutsuja fetch API:n avulla. Hyväksytty lähde määriteltiin asettamalla vastauksen otsikot (headers) ja sallitut HTTP-metodit. Esimerkkikoodissa 6 esitetään palvelinpuolen vastauksen otsikon määrittelyt.

```
app.use((req, res, next) => {
  res.setHeader("Access-Control-Allow-Origin", "https://lo-
calhost:3000");
  res.setHeader("Access-Control-Allow-Methods", "GET, POST, PUT, DE-
LETE");
next();
});
```

Esimerkkikoodi 6. Määritetään sallittu alkuperä ja metodit vastauksen (response) otsikkoon.

Esimerkkikoodissa 6 nähdään alkuperän ja metodien määrittelyt otsikkoon res.setHeader-metodilla. Tässä tapauksessa vain osoitteesta ”https://localhost:3000” saapuvat pyynnöt saavat käyttöoikeudet palvelimelle ja HTTP-metodeille GET, POST, PUT ja DELETE. Palvelimelle oikeasta alkuperästä saapuvat henkilötiedot tallennetaan omaan kansioon ja yksilöidään kellonajan, päivämäärän ja henkilön nimitietojen mukaan.

POST-metodia käytetään käyttäjärajapinnassa esimerkkikoodissa 7 nähtävässä submitData-funktiossa, joka käyttää asynkronista fetch-funktiota.

```
***api.js
const submitData = async (formData, url) => {
    try {
      const response = await fetch(url, {
        method: "POST",
        body: formData,
      });
      return response;
    } catch (error) {
      throw error;
    }
  };
```

Esimerkkikoodi 7. Funktio submitData, joka käyttää POST-metodia lähettääkseen käyttäjän tiedot sisältävän formData-olion.

Esimerkkikoodissa 7 esitetään asynkroninen submitData-funktio, joka HTTP-metodia POST käyttämällä lähettää formData-olion. Tämä funktio tuodaan ohjelmiston muiden osien käyttöön api.js-tiedostosta. Funktio ottaa parametreina tietoja sisältävän objektin formData ja palvelimen IP-osoitteen url.

Esimerkkikoodissa 8 esitetään palvelimen IP-osoitteen määrittely sekä henkilö- ja kirjautumistietojen lähettäminen.

```
const apiUrl = "https://172.17.166.55:443";
const tiedot = {
      etunimi,
      sukunimi,
      yritys,
      email,
      puhelinnumero,
      vastuuhenkilo,
      date: formattedDate,
      time: formattedTime,
    };
    const jsonTiedot = `${etunimi}_${sukunimi}_${email}`;
    const formData = new FormData();
    formData.append("tiedot", JSON.stringify(tiedot, null, 2));
    formData.append("jsonNimi", jsonTiedot);
   try {
    const response = await submitData(formData, apiUrl);
    setLahetetty(true);
    console.log(response);
  } catch (error) {
    setEilahetetty(error.message);
    console.log(error);
  }
```

Esimerkkikoodi 8. Määritetään JavaScript-objekti tiedot ja jsonTiedot-muuttuja, joka lähetetään submitData-funktiota käyttäen palvelimelle.

Esimerkkikoodissa 8 esitetään Lomake-sivulla käyttäjän syöttämien tietojen tallentaminen JavaScript-objektiin tiedot. Samalla myös määritellään muuttuja jsonTiedot, joka koostuu etunimi-, sukunimi- ja email-muuttujien tiedoista. Tätä käytetään palvelimella JSON-tiedoston nimeämiseen. Sen jälkeen määritellään formData-olio, johon liitetään JSON-muodossa tiedot-objekti ja jsonTiedot-muuttuja. Funktiota submitData, joka nähtiin esimerkkikoodissa 7, käytetään formData-olion lähettämiseen. Funktiolle välitetään myös palvelimen osoite, joka määritellään muuttujalla apiUrl.

Esimerkkikoodissa 9 esitetään Multerin käyttämä upload.single-funktio ja Express-kehyksessä käytettävä app.post-funktio, joka käy läpi formData-olion mukana palvelimelle saapuvat tiedot.

```
***index.js
const app = express();
app.post("/", upload.single("tiedot"), (req, res) => {
  const tiedostonNimi = req.body.jsonNimi;
  const tiedot = req.body.tiedot;
  const date = JSON.parse(tiedot).date;
  const time = JSON.parse(tiedot).time;
  if (tiedostonNimi != undefined) {
    let nimi = tiedostonNimi;
    let i = 1;
    if (!fs.existsSync(`./uploads/${date}`)) {
      fs.mkdirSync(`./uploads/${date}`);
    }
    while (fs.existsSync(`./uploads/${date}/${nimi}.json`)) {
      nimi = `${tiedostonNimi}-${i}`;
      i++;
    }
    const uniqueId = getUniqueFileId(`./uploads/${date}`);
    const fileData = JSON.parse(tiedot);
    fileData.uniqueId = uniqueId;
    const updatedData = JSON.stringify(fileData);
    fs.writeFile(
      `./uploads/${date}/[${time}]_${nimi}.json`,
      updatedData,
      (err) => {
        if (err) {
          return res.status(500).send(err);
        }
        return res.status(200).send("Tiedosto tallennettu");
      }
    );
  } else {
  }
});
```

Esimerkkikoodi 9. Palvelinpuolen funktio app.post, joka käsittelee saapuvaa tietoa ja tallentaa sen JSON-tiedostoon.

Esimerkkikoodissa 9 esitetään, miten palvelimelle hyväksytysti tulevat tiedot käsitellään app.post-funktiossa, kun Express-sovelluksen polku ”/” saa pyynnön HTTP-metodilla POST. Funktio app.post saa parametreinä objektit req (request) ja res (resolve) ja funktion next. Multer-kirjasto käyttää upload.single-metodia tiedoston vastaanottamiseen. Tämän jälkeen luodaan muuttujat tiedostonnimelle, vastaanotetuille henkilötiedoille, kellonajalle ja päivämäärälle.

If-ehtolauseella tarkistetaan tiedostojärjestelmän hallintaan tarkoitetun fs-moduulin avulla, että tiedostolle löytyy nimi. Jos tiedosto on saapunut onnistuneesti, tarkistetaan kansiosta uploads löytyvät alikansiot sen varalta, että tiedoston päivämäärä täsmää jo aiemmin luotuun kansioon. Mikäli kyseessä on

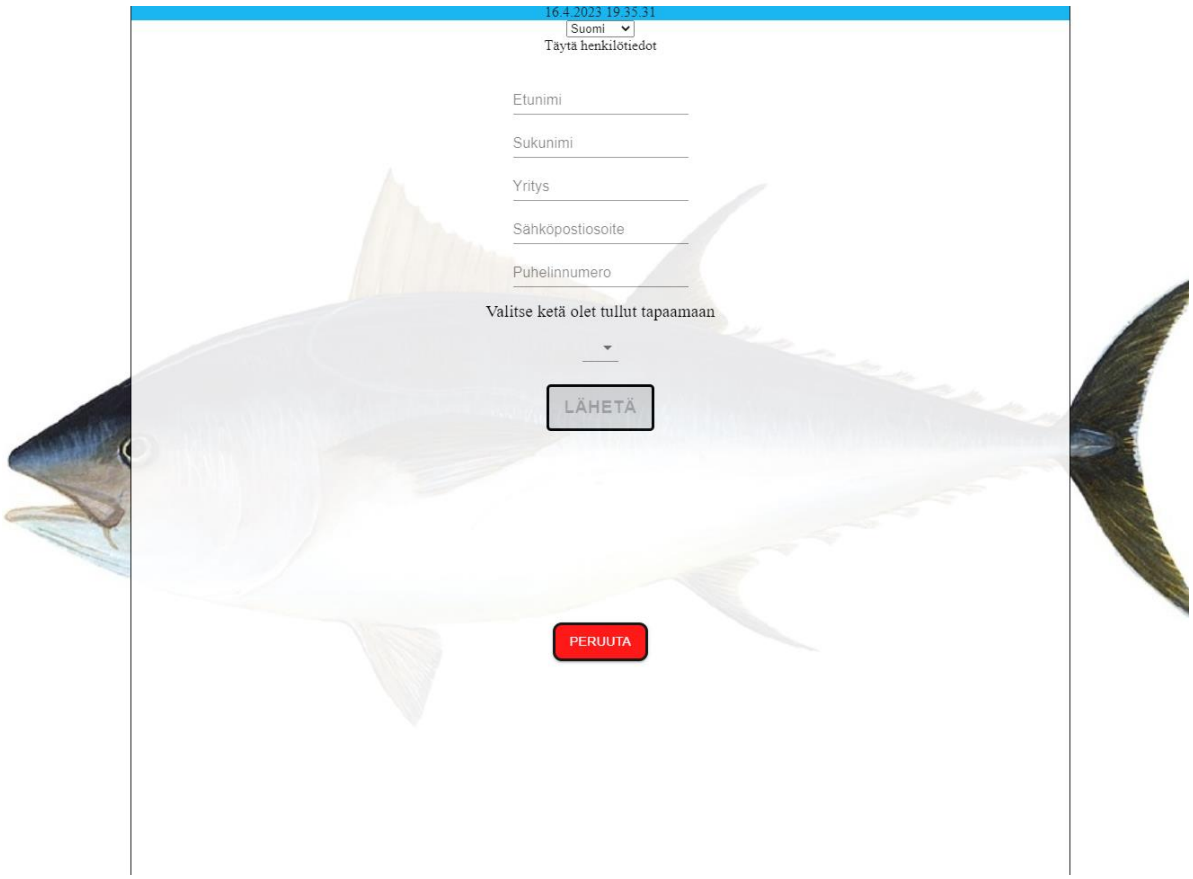
ensimmäinen lähetyksen päivämäärällä luotu tiedosto, sille luodaan oma kansio.

Sen jälkeen kaikista uploads-kansioon aikaisemmin luotujen tiedostojen lukumäärästä luodaan tiedoston yksilöivä muuttuja uniqueId ja se lisätään uutena kenttänä muiden tietojen perään JSON-tiedostoon omaksi tietokentäkseen. Tämä muuttuja määritetään funktiolla getUniqueFileId, joka käy fs-moduulin avulla läpi uploads-kansion ja sen alikansiossa olevien tiedostojen lukumäärän. Lopuksi palvelin tallentaa tiedoston tai hylkää sen ja lähettää tilan päätelaitteelle vastauksen res.status mukaan, joka on tässä tapauksessa joko 200 (OK) tai 500 (virhe).

5.4 Henkilötietojen käsittely ja tarkistus käyttäjärajapinnassa

Henkilötiedot, joihin sisältyvät etunimi, sukunimi, yritys, puhelinnumero, sähköpostiosoite ja määritelty yhteyshenkilö, syötetään Lomake-sivulla MUI:n TextField-komponenttiin ja lähetetään fetch API -rajapintaa käyttäen HTTPS-yhteyden avulla palvelimelle.

Kuvassa 3 nähdään ote Lomake-sivusta, jossa käyttäjä voi syöttää henkilötiedot ja määritellä tapaamista valvovan vastuuhenkilön.



Kuva 3. Lomake-sivu, jossa olevalle lomakkeelle syötetään henkilötiedot ja valitaan vastuuhenkilö vierailun ajaksi.

Kuvan 3 lomakesivulla käyttäjä voi syöttää henkilötietonsa. Käyttäjältä kysytään etunimi, sukunimi, yritys, sähköpostiosoite ja puhelinnumero. Lomakkeen lopussa valitaan vielä MUI:n Select-komponentilla toteutetusta pudotusvalikosta yhteyshenkilö, joka toimii myös vierailijan vastuuhenkilönä. Valikossa on mahdollisuus myös käyttää valintaa ”Muu”. Tämä valinta avaa valikon alle tekstikentän, johon käyttäjä voi syöttää omavalintaisen vastuuhenkilön nimen.

Esimerkkikoodissa 10 nähdään ote Lomake-sivulla käytetystä funktiosta handleSubmit, joka tarkistaa lomaketta lähettäessä, että lomakkeen kaikki kentät ovat täytettyjä, ja asettaa ne tilaan lomakkeenTila. Funktiota kutsutaan lomakesivulla, kun käyttäjä painaa lähetä-nappia.

```
const handleSubmit = async (event) => {
         if (!tarkistaLomake()) {
        event.preventDefault();
        setLomakkeenTila({
        ...lomakkeenTila,
        etunimi,
        sukunimi,
        yritys,
        email,
        puhelinnumero,
          vastuuhenkilo: vastuuhenkilo === "Muu" ? omaVastuuhenkilo :
          vastuuhenkilo, omaVastuuhenkilo,
        });
        return;
    }
```

Esimerkkikoodi 10. Lomakkeen tietojen asettaminen lomakkeenTila-tilaan handleSubmit-funktiota käyttäen.

Esimerkkikoodin 10 asynkronisessa funktiossa handleSubmit palaamista ei jäädä odottamaan, vaan pääohjelma voi jatkaa välittömästi suorittamista [11, s. 4]. Asynkronisten funktioiden käyttö on olennaista, jos halutaan, että ohjelmisto ei keskeytä toimintaansa odottaessaan suorituksesta palautumistaan. Ensin if-lausekkeella suoritetaan funktiolla tarkistaLomake tarkistus, että kaikki lomakkeen kentät ovat oikein täytettyjä. Mikäli funktio palauttaa arvon EPÄTOSI, asetetaan lomakkeenTila-tilaan kenttien arvot etunimi, sukunimi, yritys, email ja puhelinnumero. Vastuuhenkilö määritellään sillä perusteella, onko käyttäjä valinnut lomakkeen pudotusvalikosta löytyvän yhteyshenkilön vai määritellyt sen itse valitsemalla ”Muu” ja syöttämällä valinnaisen vastuuhenkilön nimen.

Sähköpostiosoitteen ja puhelinnumeron kentät tarkistetaan siltä osin, että sähköpostiosoite on yleisesti käytettyjen normien mukainen. Sähköpostiosoitteen täytyy sisältää vaadittavat @-merkin ja verkkotunnuksen. Puhelinnumero voi sisältää vain numeroita ja tarvittaessa +-merkin suuntanumeron esittämistä

varten. Esimerkkikoodissa 11 nähdään määritelmät sähköpostiosoitteen tarkistukselle.

```
inputProps={{
     pattern: "[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,}$",
     title: "Syötä sähköpostiosoite muodossa esimerkki@esim.fi",
}}
```

Esimerkkikoodi 11. Sähköpostiosoitteen tarkistaminen ja käyttäjälle näkyvän viestin määritteleminen.

Esimerkkikoodin 11 sähköpostiosoitteen tarkistus tehdään käyttämällä TextField-komponentin inputProps-ominaisuutta. Syötetty tieto käydään läpi pattern-ominaisuudella käyttäen apuna regular expression -kieltä (regex). Ominaisuus title määrittelee viestin, joka näytetään käyttäjälle, mikäli tekstikenttään syötetty tieto ei ole oikeassa muodossa.

5.5 Henkilötietojen suojaus tietoliikenteessä

Henkilötietoja ja muuta henkilökohtaista tietoa kerättäessä on luotettavan tallentamisen ja käsittelyn lisäksi olennaista myös tiedon turvallinen lähettäminen. Ohjelmistossa ulkopuolisten pääseminen yrityksen lähiverkkoon on estetty palomuurilla, mutta se ei kuitenkaan kokonaan estä mahdollisuutta siihen, että asiaankuulumaton henkilö voisi saada pääsyn yrityksen lähiverkkoon. HTTP-protokollaa (HyperText Transfer Protocol) käytettiin insinöörityön alussa POST-kutsujen lähettämiseen, jotta henkilötietoja sisältävä lomake voitiin lähettää palvelimelle. Sitä käytettiin kuitenkin vain testausta varten, koska siltä puuttuvan suojauksen takia se ei soveltunut arkaluontoisen tiedon lähettämiseen. HTTP-pyynnöt kulkevat verkon läpi ilmitekstinä (plaintext), mikä mahdollistaa käyttäjärajapinnan ja palvelimen välillä kulkevan tietoliikenteen seuraamisen ja siten mahdollisen ulkopuolisen tahon pääsyn käsiksi HTTP-pyyntöön sisältyvään tietoon. Tässä tapauksessa luvattoman käyttäjän olisi mahdollista päästä käsiksi lomakkeelta lähetettyihin tietoihin kaappaamalla päätelaitteen ja palvelimen välillä tapahtuvaa tietoliikennettä.

Esimerkkikoodissa 12 esitetään yksi mahdollinen palvelimelta lähtevä HTTP-vastaus, jossa nähdään muiden tietokenttien ohella lähetetty teksti ”Hello World!” [13].

```
HTTP/1.1 200 OK
Date: Wed, 30 Jan 2019 12:14:39 GMT
Server: Apache
Last-Modified: Mon, 28 Jan 2019 11:17:01 GMT
Accept-Ranges: bytes
Content-Length: 12
Vary: Accept-Encoding
Content-Type: text/plain

Hello World!
```

Esimerkkikoodi 12. Palvelimen lähettämä esimerkkivastaus HTTP-pyynnössä lähetettyyn tekstiin ”Hello World!” [12].

Esimerkkikoodissa 12 nähdään vastaus HTTP-pyyntöön, joka käyttää HTTP-versiota 1.1 ja sen HTTP-statuskoodi on 200, eli pyyntö onnistui. Date- ja Server-otsikkokenttien tiedot esittävät vastauksen lähettämisen päivämäärän, kellonajan ja palvelimen käyttämän ohjelmiston. Accept-Ranges-määrittely bytes määrittelee, että vastaus voidaan lähettää osissa. Tämä on hyödyllinen ominaisuus esimerkiksi, kun käsitellään suuria tiedostoja. Vary-tietokenttä määrittelee, että palvelin voi tarjota parhaiten sopivan version vastauksesta. Se voi esimerkiksi olla gzip-kompressoitua sisältöä, mikä tarkoittaa, että HTTP-vastaus on tiivistetty. Content-Type määrittelee vastauksen sisällön, joka on tässä tapauksessa ilmitekstiä.

HTTP-protokollalle ominaisen suojaamattoman yhteyden aiheuttaman tietosuoja-aukon korjaamiseksi valikoitui suojatun HTTPS-protokollan (HyperText Transfer Protocol Secure) käyttö tiedonsiirrossa. Se käyttää TLS- (Transport Layer Security) ja SSL-protokollia (Secure Socket Layer) salatakseen tiedonsiirron. Näitä yleisesti kutsutaan yksinkertaisemmin TLS/SSL-salaukseksi.

TLS/SSL-salauksessa käytetään avainta: julkinen avain ja yksityinen avain. Nämä avaimet toimivat parina, joiden pohjalta luodaan algoritmilla uudet avaimet, joita kutsutaan istunnon avaimiksi (session keys). Istunnon avaimet mahdollistavat salatun tietoliikenteen kahden laitteen välillä kunkin istunnon ajaksi.

Tämän salauksen avulla lähetetyt HTTP-pyynnöt ja -vastaukset eivät näy ulkopuoliselle tietoliikennettä seuraavalle osapuolelle ilmitekstinä. [12.]

Yksityinen avain sijaitsee palvelimella, ja sen kanssa täsmäävä julkinen avain jaetaan palvelimen SSL/TLS-varmenteen pohjalta kullekin laitteelle, jotka yhdistävät palvelimelle. Tästä varmenteesta käytetään yleisesti myös nimeä SSL-sertifikaatti. Laitteille luodaan jaettujen avaimien pohjalta suojattu HTTPS-istunto palvelimen kanssa. SSL-sertifikaatti on niin sanottu palvelimen identiteetti, joka varmistaa siihen yhdistävälle laitteelle, että kyseessä on täsmälleen se palvelin, johon yhteys on tarkoitus luoda [12].

Luotettavia SSL-sertifikaatteja jakavat viralliset tahot, joita kutsutaan nimellä Certificate Authority (CA). Niitä ovat muun muassa Comodo, Cloudflare ja Google. Osa tarjoaa varmenteita maksullisina, mutta niitä on saatavilla myös ilmaiseksi esimerkiksi Internet Security Research Groupin (ISRG) tarjoamasta palvelusta Let's Encrypt [13]. SSL-sertifikaatteja on saatavilla kolmella eri turvallisuustasolla, jotka määrittelevät, kuinka tarkkaan palvelimen identiteetti varmistetaan. Ensimmäisellä ja toisella tasolla varmistetaan verkkotunnuksen ja organisaation omistajuus. Kolmannella tasolla tehdään vielä laajempi taustatutkimus.

Insinöörityössä virallisen CA-tahon tarjoamaa SSL-sertifikaattia ei käytetty, koska yrityksen tietoverkkoon kuulumattomat laitteet eivät saa pääsyä asiakasrajapintaan. Vierailijat eivät käytä omaa päätelaitetta henkilötietojensa lähettämiseen, joten heidän ei myöskään tarvitse huolehtia palvelimen identiteetin varmistamisesta. Suojatun HTTPS-yhteyden luominen onnistui itse allekirjoitetulla varmenteella, joka määritettiin toimimaan palvelimen identiteettinä. Yhteyden toimivuus varmistettiin testaamalla sitä kahden lähiverkossa olevan tietokoneen verkkoselaimilla.

Itse allekirjoitetun varmenteen haittapuolena ilmeni, että jokainen testauksessa käytetty selain varoitti epäluotettavasta lähteestä. Tämä varoitus ei kuitenkaan vaikuttanut HTTPS-yhteyden muodostamiseen tai toimivuuteen. Vaikka

tavoiteltu suojattu tiedonsiirto saavutettiin insinöörityön toteutuksessa, itse allekirjoitettujen varmenteiden käyttö ei ole kuitenkaan suositeltavaa muussa kuin kehityskäytössä, koska niillä ei ole ulkopuolista varmenteenmyöntäjää eivätkä ne tästä syystä ole täysin luotettavia. Ne voivat vääriin käsiin päätyessään aiheuttaa vahinkoa niiden käyttäjälle.

5.6 SSL-sertifikaatin luonti

SSL-sertifikaatin luontiin käytettiin mkcert-työkalua, jonka avulla ohjelmiston käyttöön saatiin oma CA-juurivarmenne. Tätä varmennetta käyttämällä otettiin toimintaan paikallisesti luotetut SSL-sertifikaatit kehitysympäristössä käytettäväksi. Mkcertin asennukseen tarvittiin avuksi Chocolateyta, joka on paketinhallintajärjestelmä Windowsille. Windowsin PowerShell-komentorivillä komennolla ”choco install mkcert” asennettiin mkcert-työkalu. Mkcertin asentamisen jälkeen juurivarmenne asennettiin komennolla ”mkcert -install”.

Juurivarmenteen luonnin jälkeen luotiin SSL-sertifikaatti, johon sisältyivät julkinen ja yksityinen avain. Nämä avaimet tallentuivat pem-päätteisiin tiedostoihin. Varmenteen luomiseen on käytettävissä myös muita ohjelmia mkcertin sijaan, kuten OpenSSL [14]. Sen käyttö kuitenkin osoittautui haastavaksi ohjelmistoa kehitettäessä. Salaukseen on mahdollista käyttää eri standardeja, kuten esimerkiksi RSA:ta (Rivest-Shamir-Adleman) tai DSA:ta (Digital Signature Algorithm). Varmenteen luonnin jälkeen myös tiedostojen nimet voidaan vaihtaa tarpeen mukaan.

Toteutukseen käytetty SSL-sertifikaatti luotiin käyttämällä PKCS12-standardia. Esimerkkikoodissa 13 esitetään varmenteen luonti nimellä localhost.

```
mkcert -pkcs12 localhost
```

Esimerkkikoodi 13. Mkcert-työkalulla luodaan PKCS12-standardin mukainen avainpari ”localhost”-nimelle.

Esimerkkikoodissa 13 esitetty komento ”mkcert -pkcs12 localhost” luo avainparin, joka voidaan ottaa käyttöön, jotta HTTPS-yhteys voidaan luoda.

Varmenteen tiedostomuodoksi valittiin PKCS#12, joka sisältää yleensä yhden tai useamman varmenteen, esim SSL-sertifikaatin. PKCS#12-tiedostomuotoinen tiedosto suojataan aina oletusarvoisesti salasanalla, mikä turvaa sisällön luvattomalta käytöltä.

Esimerkkikoodissa 14 esitetään käyttäjärajapinnan käyttämästä tiedostosta package.json ja palvelimen tiedostosta index.js löytyvät määrittelyt, jotka mahdollistavat päätelaitteen HTTPS-yhteyden ja siihen tarvittavat varmenteet.

```
***package.json
"scripts": {
     "start":"
     set HTTPS=true&&
     set SSL_CRT_FILE=C:/visitorcerts/cert.pem&&
     set SSL_KEY_FILE=C:/visitorcerts/key.pem&&
     react-scripts start",
 },

***index.js
const options = {
  key: fs.readFileSync("./cert/key.pem", "utf8"),
  cert: fs.readFileSync("./cert/cert.pem", "utf8"),
};
const palvelin = https.createServer(options, app);
```

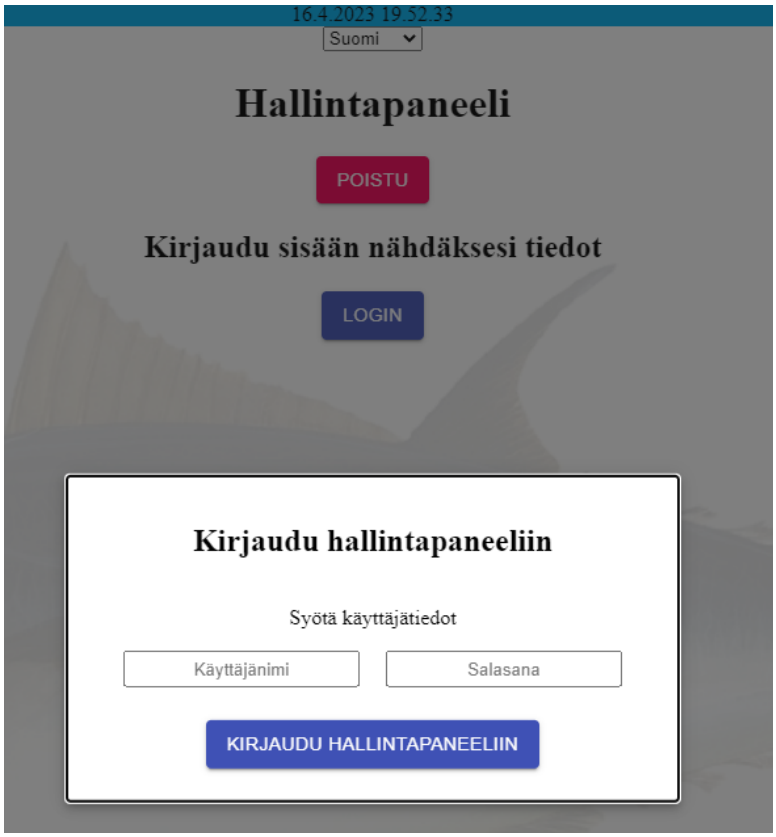
Esimerkkikoodi 14. Ote skripteistä pääteohjelman package.json:sta ja palvelimen index.js:stä, joissa määritetään käyttöön luodut SSL-sertifikaatit.

Esimerkkikoodissa 14 käyttäjärajapinnan package.json-tiedoston sisältämissä määrittelyissä HTTPS asetetaan käyttöön ja määritetään SSL-sertifikaatin käyttämät key.pem- ja cert.pem-tiedostot. Samat tiedostot määritetään myös palvelimen index.js-tiedostossa muuttujaan options, jota käytetään https.createServer-funktion toisena parametrina.

5.7 Hallintapaneeli

React Routerin osoitteeseen ”/hallinta” navigoimalla voidaan avata hallintapaneeli, jossa voi tutkia palvelimelle tallennettuja JSON-tiedostoja ja niissä olevia tietoja. Sivulla on yksinkertainen rakenne, johon sisältyvät napit poistumiseen ja

kirjautumiseen: POISTU ja LOGIN. Poistuttaessa käyttäjä palaa takaisin sivuston juureen eli kielivalintasivulle. Kuvassa 4 esitetään ote hallintapaneeli-sivulta.



Kuva 4. Ote hallintapaneelista ja siinä auki olevasta kirjautumisruudusta.

Kuvan 4 hallintapaneelissa käyttäjä voi kirjautua sisään painamalla LOGIN-nappia. Tekstikenttiin käyttäjänimi ja salasana, jotka näkyvät auki olevassa modal-ikkunassa, vaaditaan käyttäjänimi ja salasana oikeuteen tarkastella palvelimelta löytyviä henkilötietoja. Nykyisellään käyttäjänimi ja salasana ovat koodattuina ilmitekstinä ohjelmiston koodiin.

Kuvassa 5 esitetään näkymä hallintapaneelista, kun käyttäjä on kirjautunut sisään.

Kuva 5. Ote hallintapaneelissa näkyvistä card-elementeistä, joissa näkyvät vierailijoiden tiedot valitulta päivämäärältä 16.4.2023.

Kuvan 5 hallintapaneelissa näkyvät tiedot, jotka ohjelmisto hakee henkilötietoja sisältävistä JSON-tiedostoista. Hallintapaneelissa esitetään allekkain kaikki tallennetut henkilötiedot. Vaihtoehtoisesti tietoja voidaan tarkastella tietyltä päivämäärältä painamalla ”valitse päivämäärä” -kenttää ja valitsemalla sivulle avautuvasta kalenterista haluttu päivämäärä, minkä jälkeen sivulle avautuvat tiedot vierailijoista kyseiseltä ajankohdalta. Sivulla olevasta DatePicker-komponentista on valittu tarkasteltavaksi päivälle 16.4.2023 tallennetut tietokentät, jotka on syötetty testitarkoituksessa. Vastuuhenkilöt ovat peitettyjä yksityisyyden suojaamiseksi.

6 Yhteenveto

Insinöörityössä toteutettiin JavaScriptin kirjasto Reactilla ohjelmisto Ab Chipsters Food Oy:n yritysvieraiden henkilötietojen ja vierailuajankohtien käsittelyyn ja tallentamiseen. Toteutuksen tavoitteena oli saada isäntäyrityksen käyttöön yritysvieraiden henkilö- ja terveystietojen käsittelyyn ja tallentamiseen sopiva ohjelmisto mahdollisimman pitkälle kehitettynä. Sen oli tarkoitus korvata aikaisempi yritysvieraiden käytössä ollut yrityksen aulatilassa sijaitseva järjestelmä. Insinöörityössä kehitettiin vierailijahallintajärjestelmä, joka on kaksiosainen: verkkoselainpohjainen käyttöliittymä ja Node.js-ajoympäristössä toimiva yksinkertainen palvelin henkilötietojen vastaanottamisen ja käsittelemisen hallintaan. Ohjelmisto asennettiin Windows 10 -käyttöjärjestelmää käyttävälle pöytätietokoneelle, jolla myös suurin osa ohjelmiston testaamisesta suoritettiin.

Tiedonsiirto käyttöliittymän ja palvelimen välillä saatiin turvalliseksi yrityksen lähiverkossa toimivaa HTTPS-yhteyttä käyttäen, mutta henkilötietojen tallennustilaa ohjelmistoa ajavan tietokoneen kiintolevyllä ei todettu tyydyttävän turvalliseksi käyttää muuta kuin testikäytössä. Ohjelmistoa ajava tietokone ja kaksi ohjelmiston testaamisessa käytettyä tietokonetta eivät olleet tehokkaita, mutta kaikki toteutuksen ominaisuudet toimivat testattaessa nopeasti ja ongelmitta.

Toteutuksen kehittämisen aikana ongelmana ilmeni, että kaikkia tarvittavia resursseja ei ollut saatavilla aikataulun puitteissa. Sopivaa erillistä päätelaitetta ohjelmiston käyttökuntoon laittamiseksi ei löytynyt, eikä aikataulu riittänyt tyydyttävästi verkkoaseman paikallista kiintolevyä tietoturvallisemman henkilötietojen tallennustilan käyttöönottoon. Myös osa suunnitelluista ohjelmiston ominaisuuksista jäi uupumaan, kuten insinöörityön raportin kuvassa 1 nähty Lisäohjeistussivu ja rajoitetun vierailun toteutus. Myöskään henkilötietojen automaattista poistamista ei ehditty toteuttaa ohjelmistoon. Työn tuloksena saatiin kuitenkin kehitettyä toimiva ohjelmisto prototyyppinä, jota on mahdollisuus käyttää alustana jatkokehittämisessä.

Työtä tehdessä havaittiin, että yritysvieraiden hallintaan soveltuvan ohjelman kehittämisessä pitää ottaa huomioon monia seikkoja. Muun muassa henkilötietojen tietoturvallisen säilyttämisen toteutus vaatii paljon varmistamista, jotta tietoihin ei pääse kuka tahansa käsiksi esimerkiksi tietomurron kautta. Mikäli tallennettavien tietojen säilyttäminen ulkoistettaisiin, varteenotettavana ratkaisuna lopulliseen toteutukseen voisi olla mahdollisesti pilvipalvelujen käyttäminen apuna. Työn toteutuksessa käytettyjen SSL-sertifikaattien luominen ja käyttöönotto olivat ohjelmiston muiden osien kehittämiseen verrattuna työläitä. Ohjeita SSL-juurivarmenteen ja sertifikaattien luomiseen oli monia, ja ne olivat yleisesti vaikeaselkoisia tai niiden avulla HTTPS-yhteyttä ei saatu toimimaan halutulla tavalla. Itse luodut sertifikaatit aiheuttavat myös potentiaalisesti aukkoja yrityksen tietoturvaan.

Ohjelmistoa ei päästy testaamaan yritykseen kuulumattomilla henkilöillä, joten käyttökokemuksesta saadut palautteet jäivät vähäisiksi. Selvittämättä jäi, mihin ongelmiin törmättäisiin, kun ohjelmisto olisi käytännössä yritysvieraiden käytettävänä vapaasti ilman tarkempaa ohjeistusta. Virallisten tahojen kautta hankittujen SSL-sertifikaattien käyttöönottoa ja niiden toimintaa ei myöskään käyty insinöörityötä tehdessä läpi. JSON-tiedostoihin kerättyjen tietojen yksilöimiseen tarkoitetun ainutlaatuisen tunnisteen toimivuutta olisi myös pitänyt testata suuremmalla määrällä tiedostoja, joten sen toimivuutta jokaisessa tilanteessa ei saatu selville. Insinöörityön raportin kirjoittamisen ja toteuttamisen kehittämisen ajankäytön priorisoinnin takia ohjelman koodin ulkonäkö jäi kauas ohjelmoinnin standardeista, ja funktioiden ja muuttujien nimet jäivät osin epäsiistin näköisiksi. Myös osa asennetuista kirjastoista jäi lopulta käyttämättä ohjelmistossa.

Ohjelmiston jatkokehitysmahdollisuudet jäivät vielä insinöörityötä tehdessä osin tuntemattomiksi, koska sen käyttöönottoon edes yksinkertaisella tasolla vaaditaan vielä runsaasti työtä. SSL-sertifikaattien hankkiminen virallisilta tahoilta olisi olennaista, jotta tietoturvariskejä saataisiin vähennettyä. Nykyisessä muodossaan myös hallintapaneeli ilman suojattuja käyttäjänimeä ja salasanaa on tietosuojariski. Myös mahdolliset vierailijoiden kanssa tapahtuvien rajatapauksien käsittelyihin liittyvät ohjelmiston ominaisuudet, joissa tarvittaisiin

esimerkiksi rajoitettua sisäänpääsyä, vaatisivat lisää kehitystyötä, koska nykyisellään ohjelmistossa näitä toiminnallisuuksia ei ole. Ohjelmistoa voisi myös kehittää toimimaan mobiilisovelluksena React Nativella toteutettuna tai progressiivisena verkkosovelluksena (PWA).

Lähteet

1 Tietosuojalaki. 2018. 1050/5.12.2018.

2 Henkilötietojen käsittely. Verkkoaineisto. Tietosuojavaltuutetun toimisto. <https://tietosuoja.fi/henkilotietojen-kasittely>. Luettu 14.2.2023.

3 Getting started with React. 2023. Verkkoaineisto. Mozilla. <https:// https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started>. Luettu 2.4.2023.

4 Visual Studio Code FAQ. Verkkoaineisto. Visual Studio Code. <https://code.visualstudio.com/docs/supporting/faq>. Luettu 14.2.2023.

5 Brackets. Verkkoaineisto. Brackets.io. <https://brackets.io>. Luettu 21.4.2023.

6 Material UI - Overview. Verkkoaineisto. Material UI. <https://mui.com>. Luettu 2.4.2023.

7 Feature Overview. Verkkoaineisto. React Router. <https://reactrouter.com/en/main/start/overview>. Luettu 10.4.2023

8 Express. 2017. Verkkoaineisto. ExpressJS. <https://expressjs.com>. Luettu 21.4.2023.

9 Multer. Verkkoaineisto. Npm. <https://npmjs.com/package/multer>. Luettu 10.4.2023.

10 TeamViewer. Verkkoaineisto. <https://www.teamviewer.com/en>. Luettu 21.4.2023.

11 Hartiala, Henrik. 2018. Asynkronisuus JavaScriptissä. Kandidaattitutkielma. Tampereen yliopisto. Trepo-tietokanta.

12 Let’s Encrypt. Verkkoaineisto. Let’s Encrypt. <https://letsencrypt.org/>. Luettu 2.5.2023.

13 Why is HTTP not secure. 2023. Verkkoaineisto. Cloudflare. <https://www.cloudflare.com/learning/ssl/why-is-http-not-secure>. Luettu 12.4.2023.

14 OpenSSL. 2023. Verkkoaineisto. <https://www.openssl.org/>. Luettu 29.4.2023.

Esimerkki henkilötietoja sisältävästä JSON-tiedostosta

[17.43.26] teppo_testi_testi@testiyritys.fi.json
{
"etunimi": "Teppo",
"sukunimi": "Testi",
"yritys":"Testiyritys Oy",
"email":"testi@testi.fi",
"puhelinnumero":"0451234123",
"vastuuhenkilo":"Testivastuuhenkilö",
"date":"07.04.2023",
"time":"17.43.26",
"uniqueId": 0
}

Ohjelmiston lähdekoodit

https://github.com/magmake/visitorhandler

https://github.com/magmake/visitorhandlerServer