Eero Takaneva

Web-sovelluksen toteuttaminen Active Directoryllä ja moderneilla JavaScript-työkaluilla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

28.10.2021

Tiivistelmä

Tekijä:	Eero Takaneva
Otsikko:	Web-sovelluksen toteuttaminen Active Directoryllä ja moderneilla JavaScript-työkaluilla
Sivumäärä:	32 sivua
Aika:	28.10.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Ilpo Kuivanen Teknologiajohtaja Konsta Karttunen

---

Insinöörityössä toteutettiin käyttöliittymä, jossa esitetään palvelinpuolelle kerättyä tietoa. Työssä selvitettiin, miten Microsoft Windowsin hakemistopalveluita voidaan käyttää tiedon erittelyyn asiakaskohtaisesti.

Työssä suunniteltiin ja toteutettiin palvelinpuolelle tarvittavat reitit sekä toiminnallisuudet, joiden kanssa käyttöliittymä kommunikoi. Sovelluksen kirjautuminen tapahtuu Microsoft Windowsin Active Directoryllä, jonka avulla sovelluksessa esitettävät tiedot voidaan eritellä asiakaskohtaisesti. Tiedon erittelyn jälkeen työssä keskityttiin suurien tietomäärien selkeään esitystapaan ja ylläpitäjille omien toiminnallisuuksien tekemiseen.

Sovelluksen kirjautumislomakeesta lähtevä hakupyyntö käsitellään palvelimella, joka tekee LDAP-haulla käsittelypyynnön Active Directory -hakemistopalveluun. Hakupyynnöstä palautuu käyttäjälle token, jonka avulla sovelluksen tiedot eritellään asiakaskohtaisesti.

Käyttöliittymän toteutus Vue.js-kehyksellä mahdollisti suurien tietomäärien käsittelyn mutkattomasti käyttäen Vue:n eri direktiivejä. Tiedon käsittelynopeuteen vaikutti vahvasti se, että palvelinpuolelle tehdyt hakupyynnöt toteutettiin ylemällä komponentillä, josta saatu tieto välitettiin lapsikomponenteille. Lopputuloksena syntyi sovellus, jonka toiminnallisuudet ovat sulavia ja käyttäjäystävällisiä.

Avainsanat: JavaScript, Vue.js, Active Directory

Abstract

Author:	Eero Takaneva
Title:	Developing a Web-Application Using Active Directory and Modern JavaScript Tools.
Number of Pages:	32 pages
Date:	28 October 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Software Engineering
Supervisors:	Ilpo Kuivanen, Senior Lecturer Konsta Karttunen, Chief Technology Officer

---

The goal of the study was to create a user interface which can display collected data from the backend. The thesis also examines how collected data can be sorted and displayed between different customers using Microsoft Windows directory services.

The necessary routes and functionality were designed and developed to the backend with which the user interface could communicate. Signing into the application is done with Microsoft Windows Active Directory. This way the data presented in the application can be displayed customer specifically. Having explained this, the thesis focuses on handling large amounts of data received from the backend by making it more digestible. Finally in the study the administrator functionality was implemented. The applications sign in form sends a request to the backend. The backend then makes a LDAP-search to Active Directory and returns the necessary data to the user as a token.

Building the user interface with Vue.js-framework made handling large amounts of data easy using various different Vue directives. The request data handling was also done in the parent component which then sent the data to its child components. This increased the speed of the application and made transitions feel seamless. The finished product is a fast, modern and user friendly web-application.

Keywords: JavaScript, Vue.js, Active Directory

Sisällys

Lyhenteet

AD: *Active Directory.* Microsoft Windows -käyttäjätietokanta ja hakemistopalvelu.

AD DS: *Active Directory Domain Services.* Kerää tietoa käyttäjistä, tietokoneista sekä verkon resursseista. Palvelu määrittelee myös käyttäjien oikeudet sekä varmentaa valtuudet.

AD LDS: *Active Directory Lightweight Directory Services.* LDAP:n hakemistopalveluvaihtoehto ilman AD DS:n tuomia rajoitteita.

AD CS: *Active Directory Certificate Services.* Tietoturvasertifikaattipalvelu.

SaaS: *Software as a Service.* Ohjelmiston jakelumalli, jossa ohjelmistoa ylläpidetään palvelimella ja sitä tarjotaan internetin välityksellä.

LDAP: *Lightweight Directory Access Protocol.* Hakemistopalvelujen käyttöön tarkoitettu verkoprotokolla.

API: *Application programming interface.* Ohjelmointirajapinta.

PHP *Hypertext Preprocessor.* Palvelinpuolen ohjelmakoodikieli.

HMAC *Hash-based message authentication code.* Viestintodennuskoodi.

RSA *Rivest-Shamir-Adleman.* Julkisen avaimen salausalgoritmi.

XML *Extensible Markup Language.* Merkintäkielen standardi.

SPA *Single-page Application.* Web-sovellus tai sivu, mikä dynaamisesti vaihtaa nykyisen sivun tiedon uudella tiedolla ilman uuden sivun lataamista.

HTML *Hypertext Markup Language.* Avoimesti standardoitu merkintäkieli.

DOM *Document Object Model.* XML- tai HTML-liittymä, joka esittää tiedon rakenteen puuna.

JSX *JavaScript XML.* Syntaksin laajennus JavaScriptille.

SFC *Single File Components.* Vue -komponenttitiedosto.

1 Johdanto

OneViewIT on XaaSIT Oy:lle toteutettava tiedonkeruupalvelin ja käyttöliittymä, joka toteutetaan tässä insinöörityössä. Käyttöliittymän avulla helpotetaan suurien tietomäärien käsittelyä sekä mahdollistetaan ylläpitäjälle palvelinpuolen käsittely. Tavoitteeksi on asetettu lajitella käyttöliittymässä asiakaskohtaisesti palvelimelle kerätty tieto sekä antaa ylläpitäjälle mahdollisuus muokata asiakkaita ja palvelimelle kerättävää tietoa. Tavoitteisiin kuuluu myös toteuttaa hakufunktio jakelutiedoille, joiden avulla voi etsiä tarkkaa tuotetta tallennetuista tiedoista.

Ensimmäinen tehtävä on miettiä, miten käyttöliittymä kannattaisi toteuttaa. Työpöytäsovellukset ovat yleisesti suorituskyvyltään parempia ja mahdollistavat suurempien sovelluksien tuottamisen. Työpöytäsovelluksissa ei tarvitse ajatella mobiiliskaalautuvuutta, vaan voidaan keskittyä koko näytön tilan käyttämiseen. Verkkosovellukset puolestaan mahdollistavat sovelluksen käytön missä vain, kunhan on pääsy verkkoselaimeen. Modernit JavaScript-kehykset mahdollistavat myös verkkosovelluksien toteuttamisen nopeammin ja helpommin valmiiden moduulien, teknologisien standardien ja valmiiksi tehtyjen rajapintaosien avulla. Käyttöliittymä päätettiin toteuttaa verkkosovelluksena, koska halutut toiminnallisuudet eivät tarvitse suurta suorityskykyä ja internetyhteys tarvitaan palvelimen kanssa kommunikointiin.

Seuraavaksi täytyy miettiä, miten sovelluksessa esitetään kerätty tieto asiakaskohtaisesti. Palvelimessa tiedot tallentuvat samaan tietokantaan, jossa kaikkien yrityksien tiedot tallentuvat samoihin taulukoihin. Taulukoiden rivit voidaan eritellä yrityskohtaisesti, mutta tähän täytyy keksiä toteutustapa. Hyvin yleinen erittelytapa on sovelluksen sisäänkirjautumisen yhteydessä kirjata käyttäjältä tietoa, jonka perusteella erittely voidaan toteuttaa. Täytyy siis myös miettiä, miten sisäänkirjautuminen halutaan toteuttaa, että saadaan tarvittava erittelytieto. Kaksi vaihtoehtoa nousi sisäänkirjautumisen toteuttamiseen. Ensimmäinen vaihtoehto olisi toteuttaa tietokantaan taulukko, joka sisältäisi

käyttäjien kirjautumistunnukset sekä yrityksen nimen. Käyttäjätietotaulukon tekemisessä nousisi ongelmaksi uusien käyttäjien hallinta sekä mahdolliset tietoturvariskit. Käyttöliittymään ei toteuteta käyttäjien rekisteröintiä, joten tälle toiminnallisuudelle täytyisi tehdä oma pieni sovellus. Tietoturvallisuuden kannalta olisi myös tärkeää sijoittaa taulukko eri tietokannalle. Toinen vaihtoehto olisi toteuttaa kirjautuminen Microsoftin Active Directoryn (jatkossa AD) avulla. Yrityksellä on tämä palvelu jo valmiiksi käytössä, joten sen hakemistopalvelujen hyödyntäminen sisäänkirjautumisessa mahdollistaisi tiedon keskittämistä. AD -hakemistopalveluilla voidaan käyttäjätietoihin lisätä tarvittavia tietoja, joiden avulla asiakaskohtainen erittely on mahdollista. Uusien käyttäjien lisääminen onnistuisi palvelun käyttöliittymästä ja palvelu toimisi eri palvelimella kuin tietokanta, joka lisäisi tietoturvallisuutta. Näiden perusteella sisäänkirjatuminen ja asiakaskohtaisen tiedon erittely toteutetaan AD:lla.

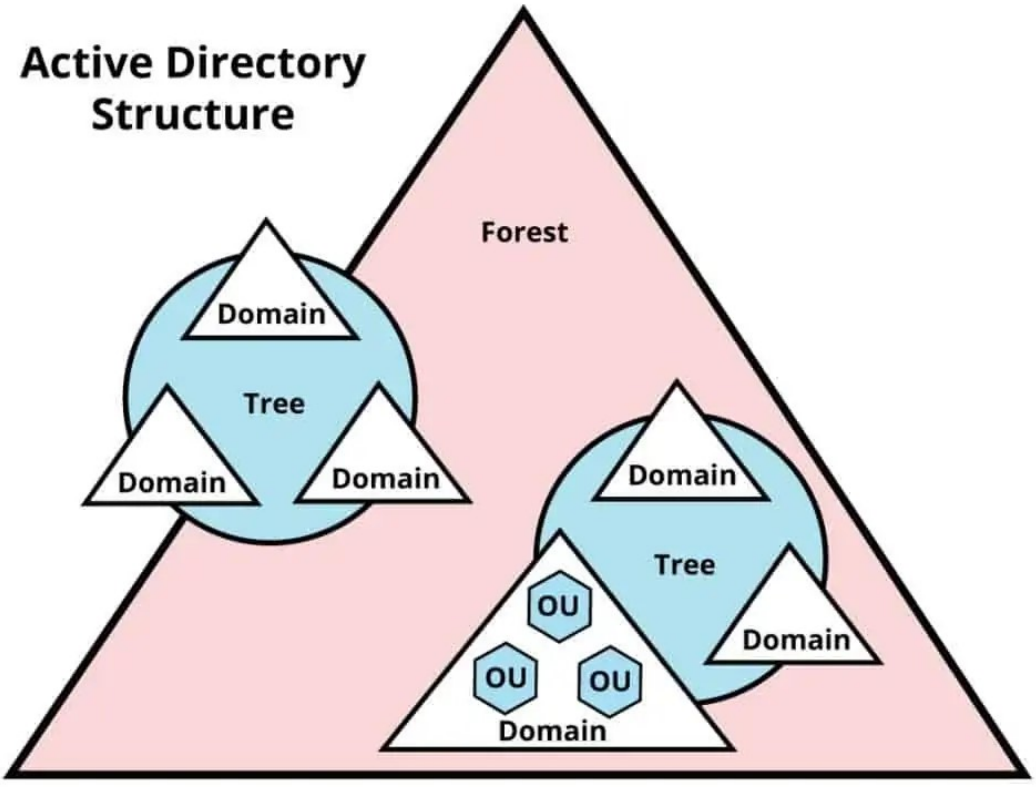
2 Active Directory

Hakemistopalveluna AD-esiintymä sisältää tietokannan ja vastaavan suoritettavan koodin, joka suorittaa hakupyynnöt sekä ylläpitää tietokantaa. AD -tietokannan olioita voidaan käsitellä muun muassa LDAP:n avulla. [1, s. 840.]

AD:n rakenne muodostuu olioilla, jotka ovat ryhmittymätietoa. Oliot jakautuvat kahteen eri laajaan kategoriaan: resursseihin, joihin kuuluvat muun muassa printerit ja tietokoneet, sekä suojausobjekteihin, jotka koostuvat esimerkiksi käyttäjistä ja jäsenryhmistä. Olio käsittelee yhtä yksikäsitteistä esim. käyttäjä-, laite- tai ryhmäesiintymää ja sen eri attribuutteja. Nämä esiintymät määritellään erikseen kaaviossa, joka määrittelee, minkälaisia olioita voidaan tallettaa AD:hen.

AD:n tietoverkoston hierarkiarakennetta voidaan tarkastella eri tasoista. AD:n organisaatioyksiköillä voidaan muodostaa toimialueen säilöttyihin olioihin arvojärjestys. Tämän avulla eri organisaatioyksikköryhmien ylläpito on helpompaa, koska tällä yleispätevällä säiliöllä voidaan eri olioluokat ryhmittää ylläpitoa varten. AD:n toimialue koostuu tietoverkoston olioista, jotka jakavat

saman AD:n tietokannan. Toimialueen ohjauskoneet mahdollistavat käyttäjien ja ryhmien todennuksen, jonka avulla voidaan hallita resursseihin pääsyä tietoverkossa. Toimialueessa luotuihin käyttäjäidentiteetteihin voidaan viitata muissa tietokoneissa, jotka sijaitsevat samassa toimialuepuuryhmässä. AD:n toimialueet, jotka noudattavat samoja loogisia rakenteita, hakemistokaaviota ja hakemistoasetuksia muodostavat toimialuepuun ja useampi toimialuepuu muodostaa toimialuepuuryhmän. Koska puut noudattavat samoja sääntöjä puuryhmien oliot ovat rakenteeltaan samanlaisia. Toimialuepuuryhmät perusasetuksiltaan luottavat kuvan 1 toimialueisiin. Kuten kuva 1 esittää, toimialuepuuryhmä on hierarkian korkeimalla tasolla ja on kokonaiskuva AD:n esiintymästä. [2; 3; 4.]



Kuva 1. AD:n rakenne [2]

Ensimmäinen AD julkaistiin Windows 1999 Server -versiossa ja tätä on päivitetty tasaisesti uusissa Windowsin palvelinversioissa. [5.] Alun perin

termillä AD tarkoitettiin vain keskitettyä nimipalvelua, mutta myöhemmin termi alkoi merkitsemään myös hakemisto- ja käyttäjäpalveluita. [6.]

2.1 Palvelut

AD palvelut koostuvat useasta eri hakemistopalvelusta. Näiden hakemistopalveluiden avulla yritykset voivat lisätä toiminnallisuuksia eri toimialuepuiden ja toimialueiden välillä. Näistä palveluista tunnetuin on Active Directory Domain Services (jatkossa AD DS), jonka toiminnallisuudet assosioidaan nykyään suoraan AD:n kanssa. AD:n hyödyntäminen palvelupohjaisen sovelluksen toteuttamisessa oli luonnollista, koska palvelu oli jo yrityksen käytössä.

2.1.1 Active Directory Domain Services

AD DS on Windows-toimialueverkon tärkeimpiä toiminnallisuuksia. Se tallettaa tietoa verkon palveluista ja infrastruktuurista sekä käyttäjistä, ryhmistä sekä jäsenistä. Toiminnallisuuksiin kuuluvat myös jäsenien pääsytietojen varmentaminen sekä oikeuksien määrittely. AD-palvelinta, jossa on AD DS, kutsutaan toimialueen ohjauskoneeksi. Ohjauskoneen tulee noudataa tiettyä kaavaa toimialuepuuryhmässä. Useat eri AD:n palvelut hyödyntävät ja käyttävät AD DS:n eri toiminnallisuuksia. [7.] Projektissa hyödynnetyt AD:n ominaisuudet ovat suurimmaksi osaksi AD DS:n toiminnallisuuksia.

2.1.2 AD Lightweight Directory Services

AD Lightweight Directory Services (jatkossa AD LDS) on LDAP -hakemistopalvelu, joka tarjoaa joustavaa tukea sovelluksille, joissa hakemistot on otettu käyttöön. Palvelu tarjoaa suurimman osan AD DS:n toiminnallisuuksista ilman tarvetta asettaa toimialueohjauskoneita tai toimialuepuuryhmiä. Tämän takia yksi tietokone voi suorittaa useampaa instanssia AD DLS, joissa jokaisessa on oma kaava. Palvelu voi myös hyödyntää AD DS:n Windows -suojausobjektin todennusta. [8.]

2.1.3 Active Directory Certificate Services

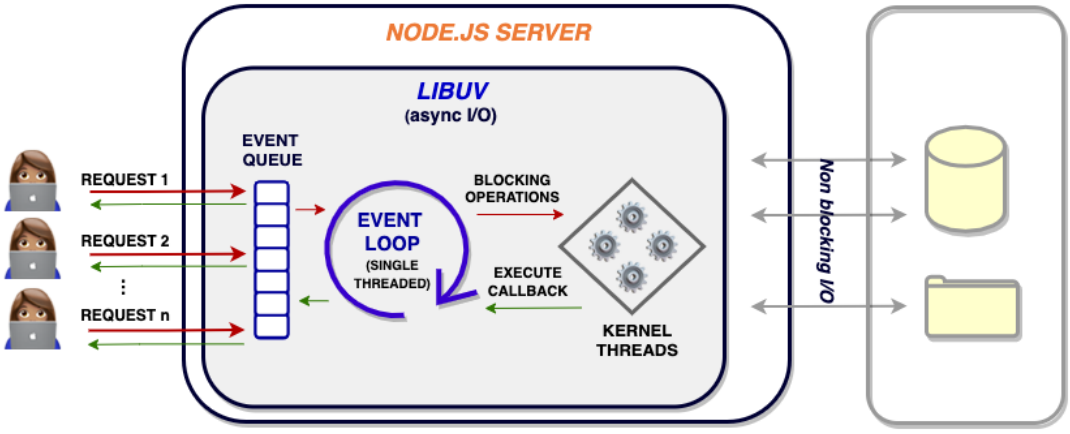
AD Certifacte Services (jatkossa AD CS) tarjoaa muokattavia palveluita julkisien avainsertifikaattien julkaisemiseen sekä hallintaan. Avaimia käytetään turvallisuusohjelmistoissa, jotka käyttävät julkisia avaimia. Organisaatioit voivat käyttää AD CS -palveluita parantamaan käyttäjien, laitteiden ja palveluiden turvallisuutta yhdistämällä nämä yksityisiin avaimiin. AD CS tarvitsee AD DS:n toimiakseen. [9.]

2.2 Azure Active Directory

Azure AD on Microsoftin pilvipohjainen identiteetti- ja pääsyoikeushallintapalvelu, jonka avulla käyttäjät voi kirjautua sisään sekä käsitellä ulkoisia- ja sisäisiä resursseja. [10.] SaaS -sovellukset, jotka hyödyntävät tiettyjä autentikaatiometodeja, voidaan integroida Azure AD:hen. Tämän avulla hallintapalvelua voi hyödyntää vielä suuremmassa skaalassa.

3 Node.js

Node.js (jatkossa Node) on avoimen lähdekoodin palvelinpuolen JavaScript ajoympäristö, joka toimii Google Chrome V8 JavaScript -moottorilla. Node -palvelin ajetaan yhdessä prosessissa tapahtumapohjaisesti ilman tarvetta luoda uusia säikeitä hakupyyntöjä suorittaessa. Suorittaessaan I/O (siirräntä) -operaatioita kuten tietokannan tai tiedostojärjestelmän käyttöä Node jatkaa toimintojaan, kunnes vastaus operaatioista palautuu. Tämä mahdollistaa useiden yhteyksien muodostamisen yhdelle palvelimelle ilman, että tarvitsee käsitellä säikeitä. Kuva 2 havainnoillistaa tätä prosessia. Web-sovelluksien asiakaspuolessa käytetään usein JavaScript-kieltä, jolloin tätä samaa kieltä voidaan käyttää myös palvelimen ohjelmoimisessa. [11.] Node tarjoaa myös kattavan API -dokumentaation, mikä liittyy funktiohin sekä olioihin. [12.] Projektissa käsitellään paljon hakupyyntöjä, ja asiakaspuoli rakennettiin Vue.js JavaScript -kehyksellä, joten Noden valinta oli perusteltua.



Kuva 2. Node.js-arkkitehtuuri [13]

Web-palvelimista vain 1,4 prosenttia koostuu Node.js -palvelimista, vaikka Node on yksi käytetyimpiä ohjelmointiajoympäristöjä. [14; 15.] Tätä voidaan selittää PHP:n ja Apachen suosiolla. Vaikka Apache on toiseksi käytetyin web-palvelin, PHP vastaa melkein 80 prosentista palvelinpuolella käytetyistä kielistä. [14; 16.]

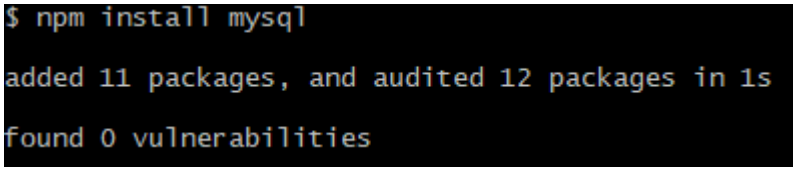
Noden kyky käsitellä funktioita ja moduuleita asynkronisesti mahdollistaa nopean ja saumattoman web-sovelluskokemuksen. PHP:n synkroninen funktioiden ja moduulien toteutusjärjestys hidastaa näiden valmistumista. Prosessorityöläissä tehtävissä Noden käyttö ei ole suositeltavaa tämän yksiprosessisen arkkitehtuurin takia. Takaisinkutsufunktioiden suuri määrä voi aiheuttaa myös epäselvää ja vaikeasti luettavaa koodia. [17; 18.]

3.1 npm ja moduulit

Noden asentajaan sisältyy npm, joka koostuu kolmesta eri komponentistä. Ensimmäinen komponentti on nettisivut, jonka avulla voi etsiä eri pakkauksia. Toinen on npm -komentorivi, jonka avulla pakkaukset voidaan ladata.

Viimeiseksi on npm -rekisteri, joka on suuri julkinen JavaScript -ohjelmien ja moduulien tietokanta. [19.]

Moduulien lataus rekisteristä on helppoa. Kuten kuvassa 3 kirjoittamalla komentoriviin: "npm install (pakkauksen nimi)" npm hakee pakkauksen rekisteristä ja asentaa sen node_modules -hakemistoon. Asennuksen jälkeen pakkauksen nimi löytyy projektin hakemistosta package.json-tiedostosta. Tärkeä on huomioida, että jos package.json-tiedostossa on asennuttamattomia moduuleita, ne voidaan asentaa kirjoittamalla komentoriviin "npm install". Tämän avulla versionhallinnassa ei tarvitse käsitellä asennettuja pakkauksia, kunhan pakkaus on merkitty package.json-tiedostoon.



Kuva 3. Moduulin asennus

Node-ympäristössä kirjastoja kutsutaan moduuleiksi. Moduulit sisältävät eri toiminnallisuuksia ja funktioita, joiden lisääminen sovelluksiin on helppoa. Moduulin lisäys sovellukseen tehdään require()-funktiolla kuten esimerkkikoodissa 1 näytetään, jonka jälkeen moduulin toiminnallisuudet ovat sovelluksen käytössä. Moduulien helppo ja mutkaton asennus on yksi Noden suurimpia vahvuuksia.

```
const express = require("express");
const app = express();
const sql = require("mysql");
```

Esimerkkikoodi 1. Express- ja sql-moduulin tuonti sovellukseen.

Projektissa on hyödynnetty erilaisia moduuleita kuten ldap-authentication-moduulia kirjautumisen suorittamiseen sekä jwt-moduulia hakupyyntöjen turvallisuuden lisäämiseksi.

3.2 Express.js

Express on Noden sovelluskehyksistä kaikista suosituin ja sitä pidetään standardivalintana verkkorajapintojen sekä verkkosovelluksien kehittämisessä. Tätä voidaan selittää Expressin kevyellä ja minimalistisella suunnittelulla, johon voidaan lisätä toiminnallisuuksia eri lisäosilla. [20.] Sovelluksia on helppo ja nopea toteuttaa, kuten esimerkkikoodissa 2 näytetään.

```
const express = require("express");
const app = express();

app.get("/", (req, res) => {
     res.send("Request received");
})
app.listen(2000, () => {
     console.log("Listening at https://localhost:2000");
})
```

Esimerkkikoodi 2. Express -sovellus, joka palauttaa vastauksen hakupyyntöön sivulla https://localhost:2000.

Tämä yksinkertaisuus voi aiheuttaa ongelmia, sillä Express tarjoaa rajoitetun virheenhallinnan, joka saattaa hankaloittaa sovelluksen kehitystä. Toinen huomattava ongelma on "X-Powered-By" -ylätunniste, jonka avulla voidaan kohdistaa hyökkäyksiä Express -sovelluksiin. Tähän ongelmaan on kehitetty moduuli, joka lisää sovelluksen suojausta, ja kyseisen ylätunnisteen voi manuaalisesti poistaa käytöstä.

3.3 JSON Web Token

Projektin palvelupohjainen rakenne toteutetaan JWT:n avulla. Tietokannassa olevat taulut ei ole tehty yrityskohtaisesti, mutta jokainen tuote ja palvelu on haettavissa yrityksen perusteella. AD:sta saatava tieto sijoitetaan tokeniin, jonka perusteella haut voidaan lajitella asiakkaalle ja taataan, ettei sovellus esitä vääriä tietoja.

JSON Web Token (jatkossa JWT) on avoimen standardin (RFC 7519) mukainen, joka määrittelee ytimekkään tavan välittää tietoa turvallisesti eri

osien välillä JSON-olioina. [21.] JWT:t voidaan allekirjoittaa HMAC:lla eli salaisella avaimella tai julkisella ja yksityisellä avaimella RSA:n avulla. JWT:n rakenne koostuu kolmesta osasta, jotka erotellaan pilkuilla. Ensimmäinen osa on "Header", joka koostuu tokenin tyypistä sekä käytetystä algoritmista kuten RSA. Tokenin seuraava osa on "payload", joka sisältää halutun välitettävän tiedon. Kolmas osa on "signature", jonka avulla varmistetaan, ettei viesti ole muuttunut matkalla ja yksityisten avainten tapauksessa varmistetaan tokenin lähettäjä. Viestin sisältö on avattavissa, joten salaisen tiedon välittäminen JWT:llä ei ole suositeltavaa. Koodiesimerkissä 3 esitetään tokenin teko onnistuneen kirjautumisen jälkeen.

```
const token = jwt.sign(
     {payload: data},
     token_key,
     {algorithm: "RSA", expiresIn: "2h"}
);
```

Esimerkkikoodi 3. JWT:n alustaminen. "Payload" sisältää viestitettävän tiedon ja "token_key" on yksityinen salausavain. Käytössä oleva algoritmi on RSA ja tehty token vanhenee kahdessa tunnissa, jonka jälkeen tokenin validointi epäonnistuu.

JWT:n valintaa projektissa voidaan perustella sen tuomien hyötyjen takia. JSON on koodattuna kooltaan pienempi kuin esimerkiksi XML-pohjaiset tiedonvälitysstandardit, jonka takia JWT toimii hyvin HTML- ja HTTP-ympäristöissä. JSON-jäsentelijät ovat myös hyvin yleisiä ohjelmointikielissä, koska se kartoittuu olioiksi.

4 Vue.js

Vue.js (jatkossa Vue) on avoimen lähdekoodin JavaScript -ohjelmistokehys käyttöliittymien rakentamiseen, jonka ensimmäinen versio julkaistiin 2014. Vue on suunnitteltu osittain käyttöön otettavaksi toisin kuin useissa muissa ohjelmistokehyksissä. Tämä tarkoittaa, että pääkirjasto keskittyy näkymä-osioon, minkä takia se on helppo yhdistää muihin kirjastoihin ja projekteihin. Vaikka pääkirjasto on keskitetty näkymään, Vue:lla voi toteuttaa moderneja SPA:itä eri työkalujen ja moduulien avulla. [22.] Vue:n vahvuuksiin kuuluu myös

sen selkeä ja käyttäjäystävällinen syntaksi, minkä takia sen suosio on kasvanut vuosien varrella tasaisesti.

Vue käyttää HTML -pohjaista mallisyntaksia, mikä mahdollistaa renderöidyn DOM:in sitomisen komponentin tietoon. Kaikki Vue:n mallit ovat käyttökelpoista HTML-kieltä, jonka voi jäsentää eri web-selaimille. Vue tarjoaa myös mahdollisuuden käyttää render -funktioita ja JSX:ää mallien sijaan. Vue:n selkeää syntaksia voidaan selittää SFC:n eli *.vue-tiedostoilla. Samassa tiedostossa on kapseloituna Vue -komponentin malli, logiikka sekä tyyli. Kuten esimerkkikoodissa 4 näkyy, osat jakautuvat omiin selkeisiin osioihinsa ja käyttäjä, jolle HTML, JavaScript ja CSS ovat tuttuja ja joka pystyy omaksumaan Vue:n komponentit helposti.

```
<template>
     <p class="paragraph"> {{ greetings }} </p>
</template>

<script>
export default {
     data() {
          return (
               greetings: "Hello"
          )
     }
}
</script>

<style scoped>
.paragraph {
     color: red;
     font-size: 20px;
     text-align: left;
}
</style>
```

Esimerkkikoodi 4. Vue komponentti, jonka template-osassa oleva paragraph-luokka esittää script-osiosta palautuvan tervehdysviestin. Style-osiossa muokataan paragraph-luokan ulkonäköä.

SPA:n toteutusta Vue:lla voi helpottaa käyttämällä Vue Routeria. Kyseisen reitittimen avulla komponenttejä voi kartoittaa, jolloin Vue Router voi renderöidä komponentit ilman, että sivua tarvitsee ladata uudelleen. Samoihin reitteihin voi samalla lisätä navigaatiosuojia, joiden avulla haluttuun reittiin reititin joko ohjaa tai estää pääsyn.

Tiedon välitys lapsikomponenteille onnistuu helposti "props":ien avulla. Tiedon välitys on esitetty esimerkkikoodissa 5. Välitetty arvo voi olla staattinen tai dynaaminen. Kun arvo on asetettu "props":ille, siitä muodostuu komponentille ominaisuus. Komponentti voi sisältää useita eri "props":eja. Tiedon välitys ylemmälle tasolle onnistuu "$emit"-toiminnon avulla.

```
<template>
     <Button text="Click me!" color="cornflowerblue"/>
</template>
<script>
import Button from "@/components/Button";
</script>
```

Esimerkkikoodi 5. Staattisen tiedon välitys Button-komponentille.

Vue:n käyttö projektiin valittiin sen käyttäjäystävällisyyden sekä vahvan skaalautuvuuden takia. Jos projektia ei kokonaan toteutettaisi Vue:na, olisi tämän perusominaisuuksien tuominen projektiin yksinkertaista. SFC:n avulla komponenttien ja sivun rakenteen toteuttaminen on intuitiviista ja helppoa, ja Vue Routerin avulla toteutettujen komponenttien vaihtaminen tuntuu luonnolliselta. Vaikka Vue:n suorituskyky ei ole yhtä tehokas kuten esimerkiksi Reactissä, Vue:n valinta voidaan perustella edellä mainituista syistä.

5 OneViewIT

5.1 Projektin tavoite

OneViewIT -projektin tavoite on tehdä asiakaspuoli aiemmin tehdylle tiedonkeräyspalvelimelle. Projektissa kirjautuminen toteutetaan AD:n avulla, joka todentaa käyttäjän sisään kirjautumisen. AD:n avulla verkkosivun materiaali lajitellaan asiakaskohtaisesti sekä varmistetaan tiedonvälityksen turvallisuus. Sovellukselle tehdään kojelautasivu, joka esittää tärkeät asiat ja tapahtumat visuaalisesti. Sivu tarjoaa asiakkaalle yleisen tietoturvallisuuden tilanteen, sekä mahdollisuuden tarkastella saatuja tietoturvailmoituksia tarkemmin. Asiakkaalle esitetään myös jakeluyritykseltä saatua tietoa, jonka avulla asiakas voi seurata tehtyjä tilauksiaan sekä varmistaa esimerkiksi

tilattujen tuotteiden takuuajat. Järjestelmän hallitsijoille palveluun toteutetaan asiakkaiden lisäämistoiminnallisuus, sekä mahdollisuus seurata kaikkea talletettua tietoa. Toiminnallisuudet halutaan toteuttaa moduläärisesti, jotta uusien toiminnallisuuksien lisääminen on sovellukseen helppoa.

5.2 Palvelinpuoli

5.2.1 Active Directoryllä kirjautuminen

OneViewIT:n projekti aloitettiin toteuttamalla palvelinpuolelle kirjautumiseen hakupyyntöreitti, joka todentaa sisäänkirjautumisen AD:n avulla. Hakupyynnössä vastaanotetaan käyttäjältä käyttäjänimi sekä salasana, jotka välitetään todennusmetodille parametreinä. Metodi käyttää saatuja arvoja ldap-authenticate moodulin asynkronisessa funktiossa authenticate(), joka palauttaa onnistuneessa kirjautumisessa käyttäjän tiedot. Jotta moduulin funktio palauttaisi käyttäjien tietoja pelkän todennuksen sijaan, täytyy LDAP-hakua laajentaa. Hakuun lisätään myös "usernameAttribute"- ja "username" -ominaisuudet, joihin voidaan käyttää kirjautujan käyttäjänimeä uudelleen kuten esimerkkikoodissa 6 esitetään. Onnistuneessa kirjautumisessa authenticate() palauttaa käyttäjän tiedot, jotka varmistetaan päteviksi validateADValues()-funktiossa.

```
async function auth(username, password) {
     const options = {
          ldapOpts: {
               url: "ldap://ldap.example.com"
          },
          userDn: `CN=${username}, CN=Users, dc=example, dc=com`,
          userPassword: `${password}`,
          userSearchBase: "dc=example, dc=com",
          usernameAttribute: "name",
          username: username
     }
     const user = await authenticate(options);
     const validUser = await validateADValues(user.memberOf, user.com-
pany);
     return {
          membership: validUser.member,
          company: validUser.company
     }
}
```

Esimerkkikoodi 6. Käyttäjätietojen todennusfunktio.

Yritys- ja jäsenryhmätiedot palautetaan moduulista hakupyyntöfunktiolle. Muille LDAP-haun palauttamille tiedoille ei ole tarvetta, mutta tulevaisuudessa palautusarvojen lisääminen on helppoa. Hakupyyntöfunktio tarkistaa, että moduulista palautuva jäsenryhmätieto sisältää arvon ’OneViewIT Users’ ja että authenticate-funktio ei palauttanut virhettä. Jos kirjautumistiedot ovat väärin ja authenticate-funktiosta palautuu virhe, asiakaspuolelle lähetettävä vastaus ilmoittaa kaikkien tietojen olevan väärin. Jos asiakkaan kirjautumistiedot ovat oikein, mutta joko yritys- tai jäsenryhmätiedoissa on puutteita, välitetään se asiakaspuolelle. Täten käyttäjä tietää, mitä oikeuksia kirjautumisessa puuttuu. Onnistuneessa kirjautumisessa oikeilla tiedoilla asiakaspuolelle lähetetään onnistunut vastaus, joka sisältää JWT:n avulla salatun yritystiedon. Palvelimelle ei ole toteutettu reittiä ja toiminnallisuutta käyttäjän luomiseksi, koska järjestelmän ylläpitäjä tekee sen AD:n avulla.

5.2.2 Tokenin todennus

Palvelinpuolen reiteille on hyvä asettaa suojaukset, ettei hakupyyntöjä voi tehdä ilman oikeuksia. Onnistuneessa kirjautumisessa käyttäjälle palautuu JWT:llä salattu token, jonka avulla hakujen teko onnistuu. Ensin tarkistetaan, että käyttäjä lähettää haussa tokenin, jonka jälkeen JWT:n tarjoamalla verify-

metodilla tarkistetaan vastaanotettu token. Esimerkkikoodi 7 esittää, miten token verrataan avaimeen ja asetettuun aikarajaan, ja jos token on pätevä, hakupyyntö voidaan suorittaa. Epäonnistuneessa validoinnissa hakupyyntö keskeytetään, eikä hakufunktioiden logiikkaa suoriteta ollenkaan.

```
const tokenVerification = (req, res, next) => {
     const token = req.body.token
     if(!token) {
          return res.status(403).send("A token is required");
     }
     try {
          const decodedToken = jwt.verify(token, TOKEN_KEY);
          req.user = decoded;
     } catch (e){
          return res.status(401.send("Invalid token");
     }
     return next();
}
```

Esimerkkikoodi 7. Hakupyyntöjen validointifunktio.

Validointifunktio asetetaan reitteihin väliohjelmaksi, jolloin se suoritetaan ennen itse oikean reitin suorittamista. Hakujen suojaaminen lisää tietoturvallisuutta sekä mahdollistaa tallennetun tiedon erittelyn asiakaskohtaisesti.

5.2.3 Hakureitit

Palvelimen keräämä tieto esitetään asiakaspuolella tokenin avulla asiakaskohtaisesti. Tieto on kerätty eri lähteistä ja ne esitetään sivuilla omissa osioissaan. Tätä varten on toteutettu useita eri hakureittejä, joihin sivusto tekee pyyntöjä. Tietoturvatiedosta välitetään omilla reiteillä hälytyksien määrä, hälytyksen aiheuttama syy sekä eri tietoturvaohjelmien tilat. Jakelutiedosta välitettävä tieto on rakennettu yhteen reittiin, koska kaikki tarvittava tieto on helposti eriteltävissä selainpuolella. Esimerkkikoodi 8 kuvaa tätä.

```
router.post("/distributions/all", auth, async function (req, res){
     const token = jwt.decode(req.body.token);
     const result = await query(`SELECT name, code, sNumber, manufac-
turer, warranty, warrantyStartTime, warrantyEndTime, tracking, refer-
ence, dispatch, product.shipment FROM data, product WHERE prod-
uct.shipment = data.shipment AND dName like "%${token.data}%";`)
     res.send(200).send(result);
     res.end();
})
```

Esimerkkikoodi 8. Reitti, joka palauttaa selainpuolelle talletetun jakelutiedon asiakaskohtaisesti.

Palvelinpuolen kaikki tarvittavat reitit luodaan ennen selainpuolen toteuttamista. Reitit viedään palvelimelle, jossa sovellus ottaa ne käyttöön. Koska reitit ovat moduläärisiä, on uusien reittien lisääminen helppoa. Tämän avulla uusien toiminnallisuuksien lisääminen onnistuu mutkattomasti ja helposti.

6 OneViewIT Vue -sovellus

6.1 Reititys

Ensimmäiseksi sovellukseen tehdään OneViewIT Vue -sovellus, jolla SPA saadaan toteutettua. Samalla sivun reitteihin voidaan asettaa "Navigation Guards", joiden avulla voidaan varmistaa, että käyttäjä kirjautuu ja että käyttäjällä on oikeudet esimerkiksi järjestelmän hallinoitsijan sivuille. Kuten esimerkkikoodissa 9 näytetään, jos käyttäjä yrittää edetä sivulle ilman kirjautumista, käyttäjä siirretään takaisin kirjautumissivulle. Samalla idealla järjestelmän ylläpitäjän sivut voidaan suojata.

```
router.beforeEach((to, next) => {
     if(to.name !== "Login" && loggedIn !== true) {
          next({ name: "Login" });
     } else {
          next();
     }
})
router.beforeEach((to, next) => {
     if(to.name === "Admin" && isAdmin !== true) {
          next({ name: "Dashboard" });
     } else {
          next();
     }
})
```

Esimerkkikoodi 9. "Navigation Guards".

Sovelluksen reitit merkitään samaan tiedostoon "Navigation Guardin" kanssa. Reitit ensin määritellään reitittimeen, jonka jälkeen ne viedään sovelluksen käyttöön otettavaksi. Reitit ovat lista olioita, jotka sisältävät reitin nimen, SFC-komponentin, johon reitti viittaa, sekä mahdolliset uudelleenohjaukset. Esimerkkikoodi 10 kuvaa tätä.

```
import Login from "@/views/Login";
const routes = [
     {
          path: "/",
          redirect: {
               name: "Login"
          }
     },
     {
          path: "/login",
          name: "Login",
          component: Login
     }
]
```

Esimerkkikoodi 10. Vue Routerin reittien määrittely. Jokaisen reitin tulisi osoittaa komponenttiin. Komponentti Login on tuotu "views"-hakemistosta, jolloin siihen voi viitata reitissä.

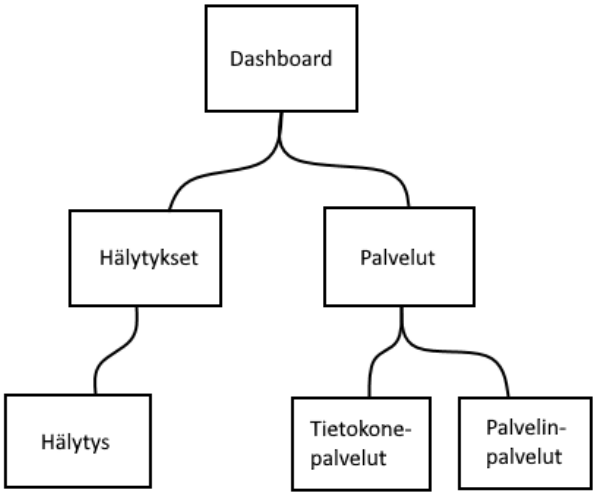
"Views"-hakemistoon tehdään sovelluksen sivuille omat komponentit, joiden avulla käyttäjä voi navigoida sovelluksen eri palveluita. Reitit esitetään sovelluksessa "router-view"-komponentilla. Jokainen "view"-komponentti sisältää saman navigointipalkki-komponentin, joka sisältää linkit sovelluksen eri "view"-komponentteihin.

6.2 Kirjautuminen

Sivustolle kirjautuminen toteutetaan seuraavaksi. Kirjautumislomake sisältää käyttäjänimi- ja salasanakentät, jotka sidotaan "input"-muuttujaan Vue:n "v-model"-direktiivin avulla. Kirjautumis-funktiossa ensin tarkistetaan, että käyttäjä on täyttänyt molemmat kentät. Tarkistuksen jälkeen funktio lähettää "input"-muuttujan tiedot "POST"-pyyntönä palvelimelle. Palvelin tarkistaa AD:sta käyttäjän kirjautumistiedot ja käyttäjätiedot. Jos käyttäjältä puuttuu ryhmä- tai yritystietoa, sovellus ilmoittaa tästä käyttäjälle. Onnistuneessa kirjautumisessa käyttäjän JWT-muodossa palautuva yritys talletetaan "sessionStorageen" ja käyttäjä ohjataan kojelautasivulle. Jos käyttäjän hakupyynnöt eivät onnistu JWT:n aikamääreen takia, uudelleen ohjaten käyttäjä takaisin kirjautumissivulle. Kirjautumisen jälkeen kaikki hakupyynnöt käyttävät saatua JWT-merkkijonoa, jotta asiakaskohtaiset haut voidaan tehdä.

6.3 Kojelauta

Kirjatumisen jälkeen toteutetaan sovelluksen kojelauta. Kojelaudan idea on esittää tärkeimmät asiat visuaalisesti, joita painamalla käyttäjä ohjautuu oikeisiin osioihin. Kojelauta-komponentti sisältää hälytys- sekä tietoturvapalvelu-komponentin, jota visualisoidaan kuvassa 4. Tulevaisuudessa uusien komponenttien lisääminen kojelaudalle onnistuu mutkattomasti.



Kuva 4. Kojelaudan jäsentely

Hälytykset-komponentti lähettää hakupyyntöjä palvelimelle, joka palauttaa hälytyksien määrän ja hälytyksien syyt. Hälytykset lajitellaan seuraavien turvatasojen mukaan: korkea, keskitaso ja alhainen. Hälytyksien määrät sijoitetaan omiin muuttujiin ja nämä välitetään yksikköhälytyskomponentille "props"ina. Yksikköhälytys-komponentti käyttää saadut tiedot ja esittää ne sivulla. Hälyksien määrät esitetään kojelaudalla, mutta näitä painaessa ohjataan käyttäjä tietoturvasivulle, jossa hälytyksien syyt näkyvät taulukossa. Sama hälytys-komponentti esiintyy siis kahdessa sivussa. Tiedon sivukohtainen lajittelu toteutetaan tarkistusfunktiolla, joka tarkistaa käyttäjän senhetkisen sivun ja siirtää tarvittaessa käyttäjän tietoturvapalvelun sivuille. Tällä tavalla kojelaudalle jätetään vain graafiset kuvaukset tiedosta.

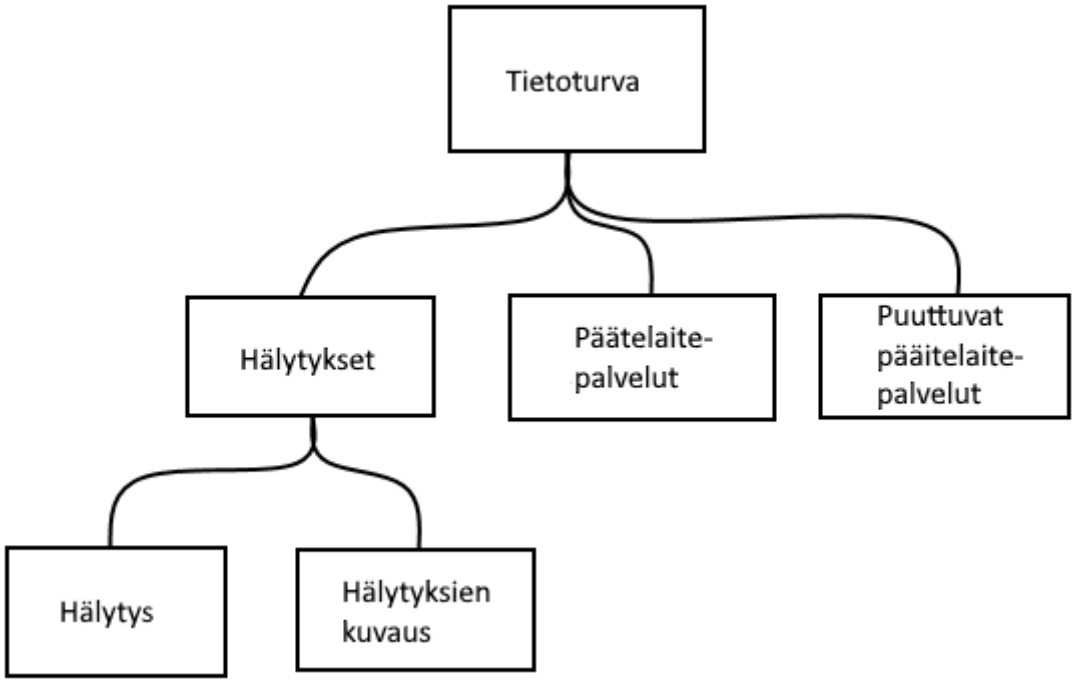
Tietoturvapalvelut-komponentti hakee palvelimelta yrityksen tietoturvaan liittyvistä päätelaitepalveluista tietoa ja välittää tiedon kahdelle lapsi komponentille. Toinen lapsikomponentti käsittelee tiedon tietokoneista ja toinen palvelimista. Lapsikomponentit lajittelevat palvelutiedoista yleisen terveys-arvon ja laskevat näiden määrän. Terveys-arvot voivat olla joko: hyvä, huono tai epäilyttävä. Arvot ja määrät sijoitetaan muuttujiin ja välitetään VueApexCharts-komponentille "props"ina. VueApexCharts muodostaa saaduista tiedoista

vuorovaikutteisen rengaskaavion, jota painamalla käyttäjä välitetään tietoturva-sivulle. Kuvassa 5 esitetään kojelaudan lopputulos.

Kuva 5. Dashboard

6.4 Tietoturva

Seuraavaksi sovellukseen toteutetaan tietoturvasivu, jossa tarkennetaan kojelaudalla esitettyjä tapahtumia. Kuten aiemmin mainittiin, painamalla kojelaudalla eri tietoturvakomponentteja käyttäjä välitetään sivulle. Sivun rakenne visualisoidaan kuvassa 6.



Kuva 6. Tietoturva sivun jäsentely

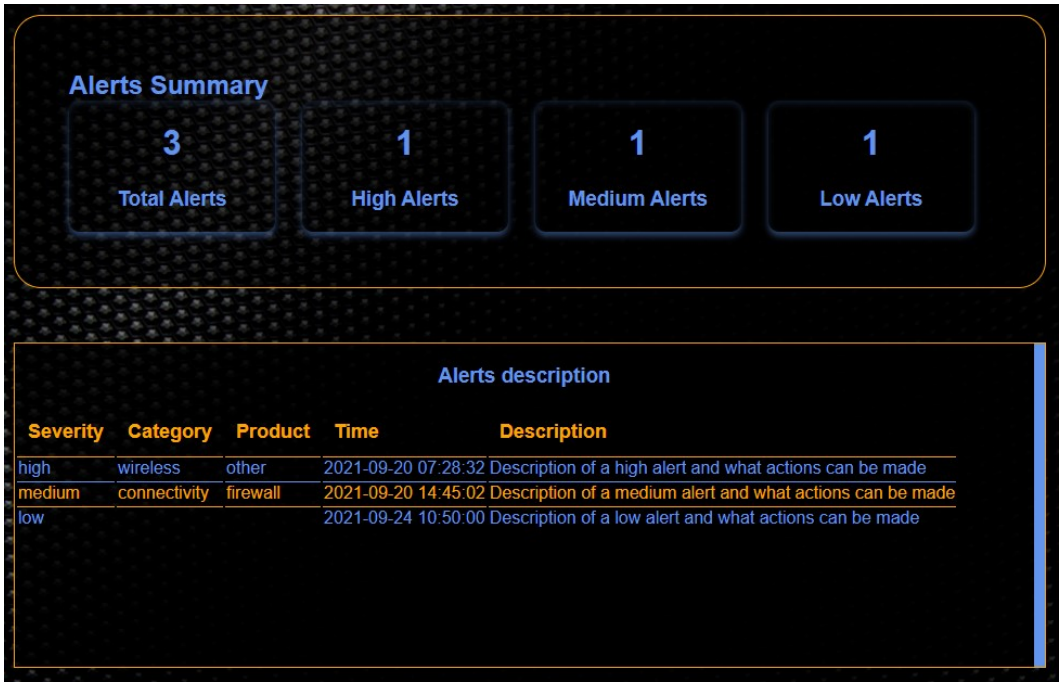
Sivu käyttää samaa hälytykset-komponenttiä kuin kojelauta, mutta ehdollisen renderöinnin avulla tarkka kuvaus hälytyksistä esitetään vain tietoturvaosiossa. Kuten esimerkkikoodi 11 esittää, "descriptions" esitetään vain, jos "correctView"-muuttuja on tosi. "CorrectView"-muuttuja on tosi, kun käyttäjä on tietoturva-sivulla.

```
<template>
     <table class="descriptions" v-if="correctView"></table>
</template>
<script>
     data() {
          return { correctView: false }
     }
</script>
```

Esimerkkikoodi 11. Ehdollinen renderöinti.

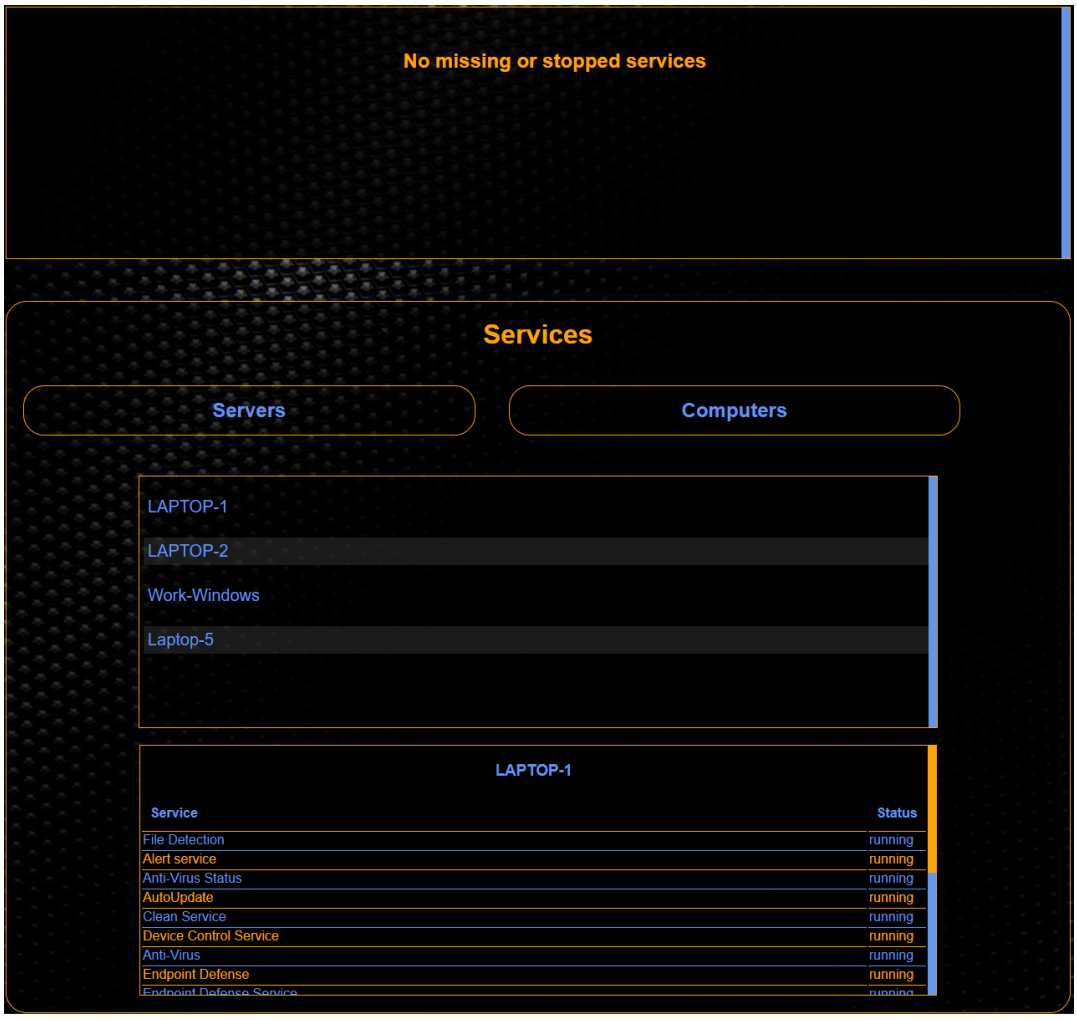
Kuten aiemmin mainittiin, hakupyynnöstä palautuu myös hälytyksien syyt. Nämä lajitellaan omiin taulukoihin turvatason perusteella. Toteutukseen halutaan mahdollistaa taulukon arvoilla lajittelu, joten Vue:n "computed"-osassa tehdään "columns()"-funktio. Funktio tarkistaa, onko hälytyksiä palautunut

hakupyynnössä ja palauttaa taulukon alkioiden laskettavien nimien ominaisuudet. "SortTable(col)"-funktio tarkistaa, onko parametrissa saadun sarakkeen perusteella lajittelu tehty. Jos taulukon sarake on jo lajiteltu, taulukko palautetaan perusmuotoon. Muussa tapauksessa taulukko lajitellaan eniten samoja alkioita omaavien olioiden mukaan. Vue:n "v-for"-direktiivin avulla taulukot kartoitetaan HTML-taulukkoon, jonka näkymää voi tarkentaa eri turvatasojen välillä. Kuvassa 7 on lopputulos.



Kuva 7. Tietoturvahälytykset

Toinen tärkeä toteuttava komponentti on tietoturvapalveluiden tilanteen tarkistus, mikä kuvattiin kojelaudalla rengaskaavioissa. Tätä varten toteutetaan 2 komponenttia: ensimmäinen esittää puuttuvat tai sammuneet päätelaitepalvelut listassa ja toinen listaa kaikkien päätelaitteiden palvelut kokonaisuudessaan. Puuttuvien tai sammuneiden palveluiden esittäminen eri osiossa mahdollistaa nopean virheen korjaamisen. Hakupyynnössä palvelin palauttaa vain päätelaitteet, joiden palvelut eivät ole toiminnassa. Komponentti-ikkuna on aina esillä, mutta ehdollisella renderöinnillä voidaan ilmoittaa käyttäjälle, ettei ole puuttuvia tai sammuneita palveluita. Kuvassa 8 on toteutetun komponentin lopputulos.

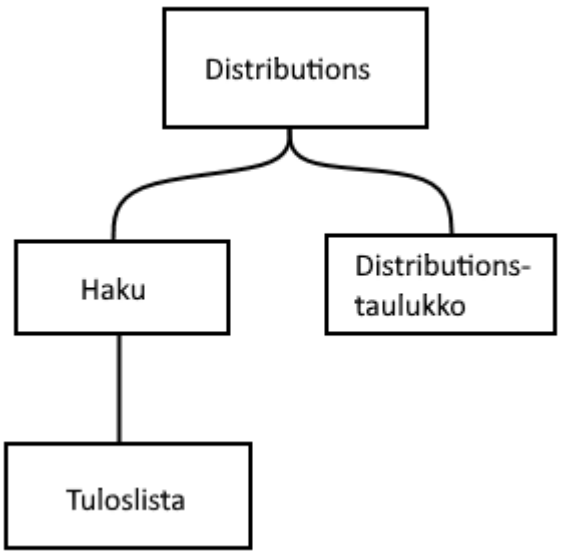


Kuva 8. Päätelaitteet ja niiden tila

Palveluiden kokonaisuutta kuvaava komponentti hakee palvelimelta kaikki päätelaitepalvelut. Päätelaitteet sijoitetaan tietokone- ja palvelinmuuttujiin. Sovellukseen tehdään tietokone ja palvelin nappulat sivulle, joilla käyttäjä voi ehdollisen renderöinnin avulla vaihtaa listanäkymää näiden päätelaitteiden välillä. Nappulaa painamalla kaikki sen tyypin päätelaitteet listataan, joita painamalla sovellus esittää kaikki päätelaitteen palvelut ja niiden tämän hetkisen tilan. Molempien päätelaitekomponenttien esitystaulukot ja lajittelu rakennetaan samalla tavalla kuin hälytyksissä. Kuvassa 8 on palvelukomponentin kokonaisuus.

6.5 Distributions

"Distributions"-sivussa esitetään jakeluyrityksiltä saatuja tietoja. Palvelimelta saadut jakeluyrityksien tiedot esitetään kahdessa eri komponentissa. Ensimmäinen komponentti on hakujentekokomponentti, jonka avulla käyttäjä voi hakea tarkasti eri tietoa. Toinen on yleinen taulukkokomponentti, joka esittää kaikki käyttäjän tiedot taulukossa. "Distributions"-komponentti tarkistaa validointifunktiolla tokenin, jonka onnistuessa komponentti hakee palvelimelta jakeluyritystiedot. Jos validointifunktio palauttaa epäonnistumisen, käyttäjä ohjataan takaisin kirjautumissivulle. Palvelimelta saadut tiedot sijoitetaan taulukkoon, josta se välitetään taulukkokomponentille ja hakukomponentille ”props”:ina. Sivun alustuksessa piilotetaan myös hakufunktion palauttama tuloslista. Sivun rakenne visualisoidaan kuvassa 9.



Kuva 9. Distributions-sivun jäsentely

Hakukomponentti on syötekenttä, jota hiirellä napsauttamalla taulukkokomponentti piilotetaan sivulta ehdollisella renderöinnillä. Napsautuksessa kutsutaan closeData()-funktioita, joka lähettää ylemmälle tasolle viestin taulukkokomponentin piilottamisesta. Hakukenttään kirjoittaessa funktio Search(e) ottaa parametrin e kohdearvon ja muuttaa arvot pieniksi kirjaimiksi, jotka sijoitetaan muuttujaan "searchString". Kyseisen merkkijonon

avulla suodatetaan kaikki jakeluyrityksien tiedot ja luodaan uusi taulukko arvoilla, jotka sisältävät merkkijonon. Tuloslistan elementti vaihdetaan näkyväksi ja luotu taulukko sijoitetaan parametriksi funktioon displayData(filteredData). Jos parametrin e koodi on askelpalautin, välitetään ylemmälle tasolle viesti taulukkokomponentin näyttämisestä sekä piilotetaan tuloslistaelementti. Koodiesimerkki 12 esittää hakufunktion käyttöä, jossa haun voi tehdä tuotteen sarjanumerolla tai tuotenimellä. DisplayData()-funktiossa tehdään uusi taulukko, jossa parametrissä välitetyt arvot asetetaan html-lista rakenteeseen, ja sijoitetaan nyt näkyvään tuloslistaelementtiin. Funktio tarkistaa samalla, että sarjanumero on muu kuin nolla. Jos arvo on nolla, arvo muutetaan ilmoittamaan, ettei tuoteella ole sarjanumeroa. Koodiesimerkkissä 12 näytetään myös, kuinka funktio palauttaa listan, joka sisältää tuotteen nimen, koodin sekä sarjanumeron. Lista sijoitetaan tuloslistaelementtiin.

```
async Search(e) {
     const searchString = e.target.value.toLowerCase();
     this.$emit("showData", false);
     if(e.code === "Backspace") {
          document.getElementById("resultList").style.display = "none";
          this.$emit("showData", true);
     } else {
          let filteredData = this.results.filter((detail) => {
          return (
          detail.name.toLowerCase().includes(searchString) ||
          detail.serialnumber.toLowerCase().includes(searchString)
          )})
          document.getElementById("resultList").style.display = "block"
          await this.displayData(filteredData);
     }
}
async displayData(deets) {
     const htmlString = details.map((detail) => {
          if(detail.serialnumber === "NULL")
               detail.serialnumber = "No serial number";
     return
     `<li>
          <p>Name: ${deets.name}</p>
          <p>Code: ${deets.code}</p>
          <p>Serial number: ${deets.serialnumber}</p>
     </li>`;
     }).join("");
     document.getElementById("resultList").innerHTML = htmlString;
}
```

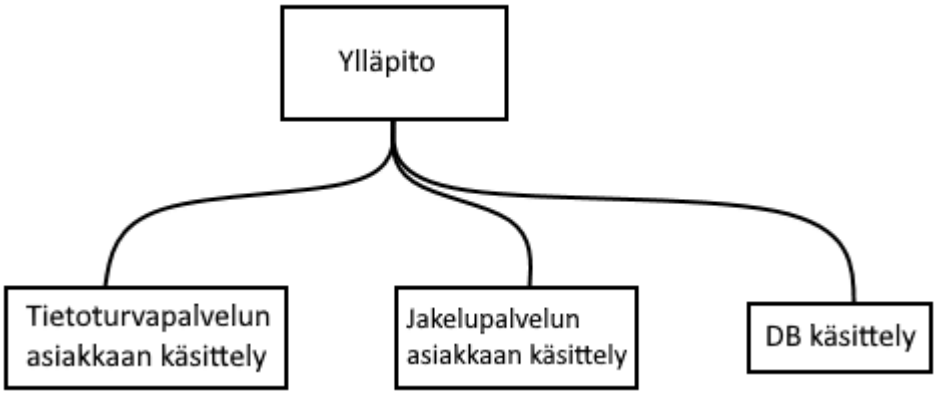
Esimerkkikoodi 12. Hakukomponentin haku- ja esitysfunktiot.

Alkuperäisessä suunnitelmassa kokonaisuus taulukkokomponentti oli tarkoitus toteuttaa kolmella eri taulukolla. Ensimmäinen taulukko käsittelee tuotteiden nimiä, koodeja sekä sarjanumeroita. Toisessa taulukossa esitetään sarjanumerot, nimet sekä takuuajat, ja kolmannessa olisi tilauksien seuranta. Kolmen taulukon dynaaminen täyttäminen eri osilla samoja tieto-osia oli selaimelle raskasta, jonka takia päädyttiin toteuttamaan yksi taulukko. Taulukko sisältää kaikki aiempien taulukoiden tiedot, ja projektin edetessä taulukkoon lisättiin uusia tietosarakkeita.

Taulukko luotiin samoilla periaatteilla kuin aikaisemmatkin. Jos hakupyyntö palvelimelle ei palauta tietoa, esitetään taulukon sijaan, ettei ole rekisteröityjä tuotteita. Taulukko käyttää samaa lajittelumetodia kuin aiemmat toteutukset, mutta taulukon tiedot on kirjoitettu ohjelmaan yksitellen. Tällä yritetään helpottaa tulevaisuudessa mahdollisien uusien tietosolujen lisäämistä taulukkoon.

6.6 Ylläpito

Viimeisenä komponenttina toteutetaan järjestelmän ylläpitäjille sivu. Sivu koostuu kahdesta asiakkuuksien lisäys- ja muokkauskomponentista. Koska sivu tulee sisältämään tietoturvallisesti tärkeää tietoa, varmistetaan reitittimen ja AD:n avulla, ettei sivua voi käyttää muut kuin ylläpitäjät. Rakenne on visualisoitu kuvassa 10.



Kuva 10. Ylläpito-sivun jäsentely

Ensimmäinen asiakkuuskomponentti käsittelee tietoturvapalveluita. Komponentin tulee mahdollistaa asiakkuuksien lisäys, muokkaus sekä poisto. Palvelinpuolelle tehdään reitit näiden toteuttamiseksi. Myös tietokantaan lisätään tarvittavat taulukot asiakastiedoille. Komponentin avulla ylläpitäjä voi lisätä asiakkaan tietoturvatiedonhakuun ilman manuaalista lisäystä palvelinpuolella.

Komponentti koostuu kahdesta nappulasta: asiakkuuden lisäyksestä sekä asiakkuuden muokkauksesta. Asiakkuuden lisäys -nappulalla avataan ehdollisella renderöinnillä lomake, jossa tarvitaan käyttäjän uudelleenkirjatumisen lisäksi haltijan tunniste, käyttäjätunniste sekä käyttäjäsalaisuus. Lomakkeen syötteet sijoitetaan muuttujiin "v-model" -direktiivin avulla, jonka jälkeen palvelimelle tehdään kaksi hakupyyntöä. "V-model" -direktiivin käyttö esitetään koodiesimerkkissä 13. Ensimmäisessä haussa varmistetaan uudelleenkirjatumisen onnistuminen, jonka jälkeen lähetetään hakupyyntö uuden tietoturva-asiakkuuden lisäyksestä. Onnistuneesta hakupyynnöstä palautuu viesti käyttäjälle, ja palvelin aloittaa tiedon keräämisen uudelle asiakkaalle.

```
<form>
     <label>
     <input type="text" name="haltija" v-model="input.haltija" place-
holder="Haltija">
     <label>
     <label>
     <input type="text" name="secret" v-model="input.secret" place-
holder="Secret">
     <label>
</form>
<script>
export default {
     data() {
          return { input: { haltija: "", secret: "" } }
     }
}
</script>
```

Esimerkkikoodi 13. Lomakesyötteiden sitominen muuttujiin.

Tietoturva-asiakkuuksien muokkasta varten varmistetaan ensin käyttäjän kirjautumisen tiedot, jonka jälkeen taulukkoon sijoitetaan hakupyynnöstä palautuneet asiakastiedot. Taulukon solujen loppuun lisätään muokkaus-

nappula, jota painamalla avautuu ehdollisella renderöinnillä uusi lomake. Lomake on valmiiksi täytetty solun alkuperäisillä tiedoilla, joita ylläpitäjä voi muokata. Lomakkeen lopussa on nappulat tietojen päivitykselle sekä asiakkuuden poistamiselle. Asiakkuuden muokkaamisen jälkeen sivulla esitettävä taulukko päivitetään nykyiseen tilanteeseen. Esimerkkikoodissa 14 on asiakkuuden päivitysfunktio.

```
async updateClient() {
     const options = {
          method: "PUT",
          headers: { "Content-type": "application/json" },
          body: JSON.stringify({
               haltija: this.input.haltija,
               asiakas: this.input.asiakas,
               secret: this.input.secret
          })
     };
     const response = await fetch("https://page/update", options);
     const data = await response.json();
     this.asiakasLista = [];
     for(let i in data.tulos) { this.asiakasLista.push(data.tulos[i]) }
     alert(data.viesti);
}
```

Esimerkkikoodi 14. Asiakkuuden päivitysfunktio.

Seuraava asiakkuuskomponentti käsittelee jakeluyritystietoja. Samoin kuin aikaisemmassa asiakkuuskomponentissa komponentin tulee mahdollistaa asiakkaan lisäys, muokkaus ja poisto. Tarvittavat reitit tehtiin palvelimelle ja palvelimen jakelutiedon hakufuntiota päivitettiin. Komponentin avulla ylläpitäjä voi muokata sovelluksessa palvelimelle tallentuvia jakelutietoja.

Komponentti toteutetaan samalla tavalla kuin ensimmäinen asiakkuuskomponentti. Uuden käyttäjän lisäyksessä ja muuntelussa tarkistetaan käyttäjätiedot, jonka jälkeen operaatiot voidaan suorittaa. Toisin kuin aikaisemmassa komponentissa jakelutiedot tarvitsevat vain yrityksen nimen, jonka avulla palvelin lajittelee, mitkä tiedot tallennetaan. Asiakkuuden poistotilanteessa välitetään palvelimelle samalla viesti, jolloin tietokannasta poistetaan tallennetut asiakkaan tiedot.

6.7 Sovelluksen ulkonäkö

Sovelluksen ulkonäön toteuttamiseen hyödynnetään Vue:n SFC -rakennetta. Jokaisen sivun ja komponentin tyyliosio on rajattu, jonka avulla sivun ulkoasun käsitteleminen pysyy selkeänä. Tumma taustakuva asetetaan kaikille sovelluksen sivuille reitittimessä ohjauksen yhteydessä.

Koska sovelluksen tausta on tumma ja sisältää paljon eri taulukoita ja kuvaajia, käytettävien värien kuuluu erottua selkeästi. Kokeilun jälkeen sivulla käytettävät yleisvärit ovat ruiskaunokin sininen sekä oranssi. Jokaisen taulukkokomponentin taustaväri asetetaan tummaksi ja lisätään hieman läpikuultavuutta, jotta komponentit erottuvat selkeämmin taustakuvasta. Koodiesimerkissä 15 näytetään, kuinka parillisten solujen värit vaihdetaan taulukossa ja tausta vaalennetaan. Yläviitteen sekä navikointipalkin pohjat muutettiin lasimaisiksi asettamalla pohjalle valkoinen väri ja nostamalla läpikuultavuutta.

```
<style scoped>
tbody tr:nth-child(2n) td {
     color: cornflowerblue;
     background-color: rgba(255, 255, 255, 0.1);
}
</style>
```

Esimerkkikoodi 15. Rajattu komponentin tyyliosio

Suurien taulukoiden takia sovelluksessa ei keskitytty suuremmin mobiiliskaalautuvuuteen. Sovellukseen lisättiin Bootstrap, joka on CSS-kehys skaalautuvien ja mobiilisivujen kehitykseen. Kehyksen avulla sivun asettelu pysyy selkeänä, vaikka näytön koko laskee.

6.8 Jatkokehitys

Työn edetessä huomattiin useita eri kohtia, joita voi myöhemmin muokata ja parannella. Ensimmäinen jatkokehitysidea on toteuttaa sovellukseen selkeä mobiiliskaalautuvuus. Suuri osa netin selailusta tapahtuu mobiililaitteilla, joten tämän ominaisuuden toteuttaminen lisäisi sovelluksen käytettävyyttä. Vaikka

sovelluksen rakenne ei ole mobiiliympäristöön tarkoitettu, uusien CSS-kehyksien avulla nämäkin voidaan toteuttaa mobiilikäyttäjille toimiviksi.

Seuraava tärkeä jatkokehitysidea olisi lisätä työssä käyttöön TypeScript. TypeScript on Microsoftin kehittämä ja ylläpitämä ohjelmointikieli, joka on tarkka syntaktinen ylijoukko JavaScriptistä. TypeScriptin tuomiin ominaisuuksiin kuuluu esimerkiksi tyyppiannotaatioiden lisääminen eri toiminnallisuuksiin, kuten luokkiin ja funktioihin. Esimerkkikoodissa 16 verrataan TypeScriptin ja JavaScriptin eroa.

```
Async function JavaScriptLaske(eka, toka) {
     return eka + toka;
}
Async function TypeScriptLaske(eka: number, toka: number): number {
     return eka + toka;
}
```

Esimerkkikoodi 16. Ylempi JavaScript-funktio sisältää kaksi parametriä, joiden tyyppiä ei ole määritelty, ja palauttaa näiden yhteenlasketun arvon. Alempi TypeScript-funktio toimii samalla tavalla, mutta parametrit on tyypitetty numeroiksi ja laskufunktion palautus on myös tyypitetty numeroksi.

Koska TypeScript on JavaScriptin ylijoukko, JavaScript -ohjelmat ovat valideja TypeScript -ohjelmia. Palvelimet kuten Node ja selain eivät pysty lukemaan suoraan TypeScriptiä, jonka takia se muunnetaan JavaScriptksi kääntäjien kuten Babelin avulla. TypeScriptin tuomien toiminnallisuuksien avulla voidaan vähentää ohjelmavirheitä huomattavasti. Moderneissa web-sovelluksissa on usein käytössä TypeScript sen tuomien etujen takia, ja uskon, että toteutetun sovelluksen laatu parantuisi huomattavasti tämän lisäyksestä.

Toisia selkeitä jatkokehitysideoita olisi toteuttaa LDAP-haku ilman ulkoista moduulia, jonka avulla voidaan lisätä haun tietoturvallisuutta. Ylläpitäjille voisi toteuttaa erillisen oman sovelluksen, jolla asiakastietoja voisi käsitellä. Tämä eristäisi ylläpito-sivun muusta sovelluksesta, joka lisäisi tietoturvallisuutta.

7 Yhteenveto

Tehdyllä työllä pyrittiin toteuttamaan käyttöliittymä, jolla voi tarkastella palvelimelle kerättyä tietoa. Tavoitteena oli toteuttaa asiakaskohtainen erittely sovelluksessa sekä antaa ylläpitäjille mahdollisuus muokata asiakkuuksiin liittyviä tietoja.

Asiakaskohtainen erittely toteutettiin käyttämällä kirjautumisessa AD:n hakemistopalveluita. Kirjautuminen todennetaan LDAP-haulla, joka palauttaa AD:sta käyttäjän tiedot. Puutteellisilla oikeuksilla kirjautumisesta ilmoitetaan käyttäjälle, kuten myös virheellisillä kirjautumistiedoilla. Onnistuneessa kirjautumisessa palautuu käyttäjän yritys, jonka avulla voidaan eritellä käyttöliittymä asiakaskohtaisesti. Uusien käyttäjien lisääminen tehdään AD:ssa eikä sille ollut tarvetta tehdä toiminnallisuuksia. Palvelinpuolelle tehtiin hakureitit, joiden avulla käyttöliittymä hakee palvelimelta tietoa. Kirjautumisen ja hakureittien turvallisuutta varten otettiin käyttöön JWT, jonka avulla tarvittavat tiedot salattiin ja haut validoidaan.

Käyttöliittymä toteutettiin Vue:lla. Sen reititys toteutettiin Vue Routerilla, jolla mahdollistettiin SPA:n tekeminen sekä luotiin "Navigation Guards". Ensimmäinen tehty sovelluksen osa oli kojelauta, jossa keskitytään näyttävästi tiedon esittämiseen rengaskaavioiden ja taulukoiden avulla. Sivun eri osioiden painaminen välittää käyttäjän tarkennetuille sivuille, jotka esittävät kattavamman kuvauksen tiedosta. Ensimmäinen kattavamman tiedon sivu liittyy tietoturvallisuuteen. Sivussa esitetään asiakkaan tietoturvapalvelun keräämä tieto sekä tietoturvapalveluiden tila. Toisessa sivussa esitetään jakeluyritykseltä kerättyä tietoa. Sivu sisältää yhden ison taulukon, joka sisältää kaikki asiakkaan tuotteet sekä hakuosion, jonka avulla voi tuotteen esimerkiksi sarjanumeroilla hakea tuotteita. Molemmissa kattavan tiedon komponenteissa sijaitsevat dynaamiset taulukontekofunktiot voitaisiin tehdä suorituskyvyllisesti paremmin sekä hakufunktion palauttama lista voitaisi muuttaa myös taulukoksi.

Ylläpitäjille tehtiin oma komponentti, jonka avulla voidaan muokata asiakastietoja sekä muokata tallennettua jakeluyritystietoa tietokannassa. Ylläpitäjäsivun komponentit ovat toiminnallisuuksiltaan hyvin samanlaiset, joten jatkokehityksen kannalta olisi hyvä muuttaa näiden toiminnallisuudet modulaarisemmaksi.

Työn tavoitteet saavutettiin sekä luotiin sovellus, joka on graafista viimeistelyä vaille valmis julkaistavaksi. Tekninen osaaminen kasvoi työn aikana huomattavasti sekä kyky toteuttaa ohjelmistoja alusta loppuun.

Lähteet

1 Mark E. Russinovich, David A. Solomon. 2004. Microsoft Windows Internals (4th Edition): Microsoft Server 2003, Windows XP, and Windows 2000.

2 PCWDLD.com. Verkkoaineisto <https://www.pcwdld.com/active-directory-guide#wbounce-modal> Luettu 1.9.2021.

3 Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc978003(v=technet.10)?redirectedfrom=MSDN> Luettu 1.9.2021.

4 Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/understanding-the-active-directory-logical-model> Luettu 1.9.2021.

5 Tools4Ever. Verkkoaineisto. <https://www.tools4ever.com/glossary/what-is-active-directory-ad/> Luettu 2.9.2021.

6 Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/previous-versions/technet-magazine/cc160894(v=msdn.10)?redirectedfrom=MSDN> Luettu 2.9.2021.

7 Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc731053(v=ws.10)?redirectedfrom=MSDN> Luettu 2.9.2021.

8 Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc754361(v=ws.10)> Luettu 2.9.2021.

9 Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc731564(v=ws.10)> Luettu 2.9.2021.

10 Microsoft. Verkkoaineisto. <https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis> Luettu 2.9.2021.

11 NodeJS. Verkkoaineisto. <https://nodejs.dev/learn/introduction-to-nodejs> Luettu 3.9.2021.

12 NodeJS. Verkkoaineisto. <https://nodejs.org/en/docs/> Luettu 3.9.2021.

13 Scout APM. Verkkoaineisto <https://scoutapm.com/blog/nodejs-architecture-and-12-best-practices-for-nodejs-development> Luettu 3.9.2021

14 W3Techs. Verkkoaineisto. <https://w3techs.com/technologies/overview/web_server> Luettu 4.9.2021.

15 Stack Overflow. Verkkoaineisto. <https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-all-respondents> Luettu 4.9.2021.

16 W3Techs. Verkkoaineisto. <https://w3techs.com/technologies/overview/programming_language> Luettu 6.9.2021.

17 Simform. Verkkoaineisto. <https://www.simform.com/blog/nodejs-vs-php/> Luettu 6.9.2021.

18 Void Canvas. Verkkoaineisto. <https://www.voidcanvas.com/describing-node-js/> Luettu 6.9.2021.

19 npm Docs. Verkkoaineisto. <https://docs.npmjs.com/about-npm> Luettu 7.9.2021.

20 Express. Verkkoaineisto. <http://expressjs.com/> Luettu 7.9.2021

21 JSON Web Tokens. Verkkoaineisto <https://jwt.io/introduction> Luettu 6.10.2021.

22 Vue.js. Verkkoaineisto. <https://v3.vuejs.org/guide/introduction.html#getting-started> Luettu 14.9.2021.