Tampereen ammattikorkeakoulu

# Selainpohjainen etähallintatyökalu

## Case: Sandvik Mining and Rock Technology

Emil Pirinen

OPINNÄYTETYÖ
Marraskuu 2019

Tieto- ja viestintätekniikka
Ohjelmistotekniikka

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

PIRINEN, EMIL:
Selainpohjainen etähallintatyökalu
Case: Sandvik Mining and Rock Technology

Opinnäytetyö 48 sivua, joista liitteitä 2 sivua
Marraskuu 2019

---

Opinnäytetyön tarkoituksena oli kehittää selainpohjainen etähallintatyökalu kaivoskoneita tuottavalle yritykselle. Työkalu rakennettiin käyttäen moderneja ja monipuolisia sovelluskehyksiä sekä nykyaikaisimpia tekniikoita. Projektin ohessa tutkittiin mahdollisuuksia erilaisten ominaisuuksien toteuttamiseen huonojen internet-yhteyksien asettaessa erinäisiä rajoitteita liikuteltavan datan määrälle. Työkalun tarkoitus on helpottaa kaivoskoneiden ongelmatilanteiden diagnosointia etänä sekä mahdollistaa varmuuskopioiden yksinkertainen talteen ottaminen.

Etähallintatyökalu koostuu kahdesta kokonaisuudesta: käyttöliittymästä ja palvelinpuolesta. Käyttöliittymä toteutettiin nykyaikaisella ja tehokkaalla Angular-sovelluskehyksellä, joka on tarkoitettu modernien web-sovellusten kehittämiseen. Käyttöliittymän värimaailma sovitettiin asiakkaan tuotteiden kanssa yhteensopivaksi. Responsiivisuuteen kiinnitettiin myös huomiota tulevaisuuden varalle, kun käytettävät päätelaitteet kehittyvät ja näyttökoot monipuolistuvat. Palvelinpuolen toteutus käyttää myös nykyaikaista, tehokasta ja helppokäyttöistä Express.js-sovelluskehystä, joka pyörii Node.js-ajoympäristössä. Molemmat kokonaisuudet noudattavat viimeisimpiä ohjelmistoteollisuuden käytänteitä ja standardeja.

Työn tuloksena syntynyt etähallintatyökalu on aktiivisessa käytössä asiakkaan kaivoskoneissa ympäri maailman. Pääasiassa asiantuntijoiden ja insinöörien käytössä oleva ensimmäinen versio etähallintatyökalusta täyttää kaikki ennalta sille annetut vaatimukset. Jatkokehitysmahdollisuuksia on kuitenkin monia ja niiden kehittäminen jatkuu pienen testivaiheen päätyttyä. Tärkeimpiä jatkokehityksen kohteita on laajentaa työkalun laitetukea useammalle kohdelaitteelle. Ensimmäinen versio on saatavilla vasta kahteen eri valmistajan tiettyyn laitetyyppiin.

Web-sovellusten käyttämien teknologioiden kehittyessä erittäin nopeaa tahtia, ovat materiaali ja lähteet pääosin peräisin kunkin sovelluskehyksen valmistajan tai kehittäjän omilta sivuilta. Kirjallisia teoksia ei tässä työssä pystytty hyödyntämään tiedon ollessa monesti jo vanhentunutta teoksen julkaisuvaiheessa.

---

Asiasanat: web-ohjelmointi, javascript, angular, express.js

## ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
ICT Engineering
Software Engineering

PIRINEN, EMIL:
Browser-based Remote Management Tool
Case: Sandvik Mining and Rock Technology

Bachelor's thesis 48 pages, appendices 2 pages
November 2019

---

The goal of this thesis was to develop a browser-based remote management tool for a company producing mining machines. The tool was built using modern and versatile software frameworks with the latest techniques. Possibilities for different features were investigated throughout the project due to poor Internet connections setting various limits for the amount of transferable data. The remote management tool is intended to ease the diagnose process during problem investigation and make it possible to take backups of the mining machine.

The remote management tool consists out of two main parts: user interface and backend. User interface was developed with modern and powerful Angular framework, which is used to create powerful and versatile single-page web applications. Color scheme of the interface was fitted to match with customer's other products. Interface was developed with responsiveness in mind because used devices develop all the time and screen size may vary in the future. Backend was also developed using modern, powerful and easy-to-use Express.js framework that runs on Node.js runtime. Both parts follow the latest software industry practices and standards.

The resulting remote management tool is actively used on client's mining machines around the world. The first version of the remote management tool, mainly used by experts and engineers, fulfills all the predefined requirements that were specified at the start of the project. However, there are many possibilities for further development that will start after a small test phase. One of the most urgent things to improve is the support for multiple target devices as the first version only supports two devices from different manufacturers.

Technologies used with web applications are developing so rapidly that material and references used in this thesis are mostly from framework manufacturer's website and documentation. Literature could not be used in this thesis due to information being mostly outdated immediately after publication.

---

Key words: web development, javascript, angular, express.js

**SISÄLLYS**

## LYHENTEET JA TERMIT

| | |
|---|---|
| API | Application Programming Interface<br>Ohjelmointirajapinta |
| Asynkroninen | Ei-reaaliaikainen, ajallisesti riippumaton |
| CI | Continuous Integration<br>Jatkuva integraatio |
| CORS | Cross-Origin Resource Sharing<br>Mekanismi ulkopuolisten HTTP-pyyntöjen hallintaan |
| CSV, .csv | Comma-separated values<br>Tietojen tallennusmuoto, tiedostomuoto |
| Django | Pythonilla ohjelmoitava web-ohjelmistokehys |
| Ethernet | Pakettipohjainen lähiverkkotekniikka |
| Git | Eräs hajautettu versionhallintajärjestelmä |
| HTTP | Hypertext Transfer Protocol<br>Protokolla selainten ja WWW-palvelinten tiedonsiirtoon |
| IoT | Internet of Things<br>Laitteiden suorittama automaattinen tiedonsiirto |
| JSON | JavaScript Object Notation<br>Avoimen standardin tiedostomuoto |
| Kernel | Käyttöjärjestelmän ydin |
| Python | Yksinkertainen, korkean tason tulkattava ohjelmointikieli |
| RAM | Random Access Memory<br>Tietokoneohjelmien työmuisti |
| Subversion | Eräs keskitetty versionhallintajärjestelmä |
| Virtualisointi | Fyysisen laitteen jakaminen loogisiksi resursseiksi, teknisten piirteiden naamiointi |
| VNC | Virtual Network Computing<br>Protokolla tietokoneen graafiseen etäkäyttöön |
| VPN | Virtual Private Network<br>Suojattu yhteys kahden tai useamman laitteen välillä |
| XML, .xml | Extensible Markup Language<br>Tietojen tallennusmuoto, tiedostomuoto |
| XSS | Cross-Site Scripting<br>Yleinen tietoturva-aukko web-sovelluksissa |

# 1 JOHDANTO

Tietokoneiden ja kannettavien älylaitteiden määrä kasvaa järisyttävällä tahdilla koko maailmassa. Teknologia ja internet ovat kehittyneet valtavia harppauksia viime vuosikymmenten aikana. Tänä päivänä yhä useampi laite, jopa hienoimmat jääkaapit, on kytketty internetiin. Tässä alati muuttuvassa maailmassa erilaisten koneiden ja laitteiden valmistajat kokevat painetta saada myös omat tuotoksensa kytketyksi osaksi tuota massiivista verkkoa. Tätä painetta on myös kaivosteollisuuden koneita ja tuotteita valmistavilla yrityksillä.

Kaivoskoneiden ollessa jo yhteydessä internetiin, halutaan niitä hallita ja tarkkailla myös etänä. Etähallintatyökalun kehityksessä on monta vaihtoehtoa tekniikoiden osalta. Tässä opinnäytetyössä päädyttiin tuottamaan kyseinen työkalu web-sovelluksena. Web-sovelluksen etuna on sen helppokäyttöisyys ja monipuolisuus käyttäjän näkökulmasta katsottuna. Samasta työkalusta riittää yksi versio, joka toimii kaikissa selaimissa kaikilla laitteilla, joihin sellainen on asennettavissa. Työkalusta ei siis tarvitse enää tuottaa useita eri versioita eri alustoille. Tässä vaihtoehdossa säästetään huomattavasti aikaa ja rahaa ominaisuuksista lainkaan tinkimättä.

Internet-yhteydet kaivosalueilla ovat vielä tänä päivänä hyvin vaihtelevat. Joissain kaivoksissa on käytössä viimeisimmät langattomat tiedonsiirtoteknologiat ja toisissa taas tyydytään vain niihin, jotka edes jollain tapaa toimivat. Tämä asettaa projektille rajoituksia ja sitä kautta joidenkin ominaisuuksien toteutusmahdollisuuksia joudutaan miettimään hieman tarkemmin.

Työn tarkoituksena on tuottaa selainpohjainen käyttöliittymä ja sille palvelinpuoli kaivoskoneiden yksinkertaiseen ja rajoitettuun hallintaan. Työssä esitellään lukuisia erilaisia projektissa käytettyjä tekniikoita ja ohjelmistokehyksiä. Lisäksi esitellään myös käyttöliittymän ja palvelinpuolen ominaisuudet sekä niitä koskevat vaatimukset ja rajoitukset. Lopputuloksena syntyvää kokonaisuutta on mahdollista käyttää pohjana ja esimerkkinä tulevia vastaavan kaltaisia sovelluksia suunniteltaessa sekä tekniikoita valittaessa.

# 2 TAUSTATIEDOT JA PROJEKTIN ALOITUS

Teknologian kehittyessä isoin harppauksin ja IoT:n yleistyessä kasvavaa vauhtia, on myös Sandvikilla oltu tietoisia kehityksen suunnasta jo pitkään. Syksyllä 2018 istuttiin muutaman henkilön kesken kokoushuoneeseen, mistä projekti ja sitä myötä myös tämä opinnäytetyö sai alkunsa. Tavoite oli selkeä. Haluttiin helppokäyttöinen käyttöliittymä, joka ei vaadi muutoksia jo olemassa oleviin alustoihin ja ohjelmiin. Hyvin nopeasti käyttöliittymän alustaksi valikoitui selainpohjainen toteutus sen monipuolisuuden ja helppouden vuoksi; käyttäjän kun ei tarvitse asentaa mitään ylimääräisiä ohjelmia.

Toiminnallisuudet hakivat palaverin aikana hieman sijojaan, mutta lopulta niissä päästiin haluttuun lopputulokseen. Mitään maata mullistavaa, kuten koneiden ohjausta, ei ollut missään vaiheessa tarkoitus luodakaan, sillä se ei käytännön syistä ole mahdollista vielä tämän päivän kaivosolosuhteissa. Sen sijaan etähallintaohjelmasta on tarkoitus ottaa lokitiedostoja talteen, kerätä varmuuskopioita ja tutkia mahdollisia hälytyksiä tai varoituksia, joita kone on kerännyt. Näiden lisäksi tutkitaan myös mahdollisuus näytön jakamiseen VNC-protokollalla sekä integroidaan toisessa projektissa tehty selainpohjainen diagnostiikkatyökalu osaksi kokonaisuutta.

## 2.1 Sandvik Mining and Rock Technology

Sandvik Mining and Rock Technology kuuluu osaksi Sandvik Groupia (kuva 1). Sandvik Group on vuonna 1862 perustettu ruotsalainen yritys, jonka liiketoiminta koostuu nykyisin kolmesta eri toimialasta, joita edellä mainitun lisäksi ovat Sandvik Machining Solutions ja Sandvik Materials Technology. Sandvik Group työllistää kaiken kaikkiaan noin 42 000 työntekijää useassa eri maassa. Yrityksen liikevaihto on vuositasolla noin 100 miljardia Ruotsin kruunua (SEK) ja yrityksen toimitusjohtajan paikkaa pitää Björn Rosengren. (Sandvik 2019)

Suomessa Sandvikilla on toimintaa Tampereella, Turussa, Lahdessa ja Vantaalla. Sandvik työllistää Suomessa noin 2400 henkilöä, joista Tampereella työskentelee 1400. Tampereella valmistetaan maanalaisia sekä -päällisiä

porauslaitteita ja tämän lisäksi siellä sijaitsevat myös teknologian osaamiskeskus, koekaivos ja automaatiotoiminnot. Turussa työskentelee noin 650 työntekijää ja siellä valmistetaan lastauslaitteita sekä dumppereita. (Yle 2019)


KUVA 1. Sandvik Groupin logo (Sandvik Group 2019)

## 2.2 Kohdelaitteet

Projektin kohdelaitteita ovat sulautetut kosketusnäytöt, joissa pyörii hyvin riisuttu Linux-pohjainen käyttöjärjestelmä. Näyttöjä on muutamia erilaisia ja valmistajina toimivat CrossControl sekä suomalainen Epec Oy. Näyttöjen tehot vaihtelevat hieman ja niissä on erityisesti tallennustilaa hyvin rajoitetusti. RAM-muistin määrä sekä prosessoritehokkuus myös osaltaan rajoittavat ja pakottavat miettimään projektin ohjelmiston ominaisuuksia ja ratkaisuvalintoja. Kohdelaitteista löytyy myös versioita, joissa kosketusnäyttöä ei ole. Tällainen näyttö on muuten täysin samankaltainen kuin muutkin, mutta ainoana erona on, että siihen ei voida asentaa koneen ohjauksessa käytettyä käyttöliittymää.

## 2.3 Kohdeympäristö

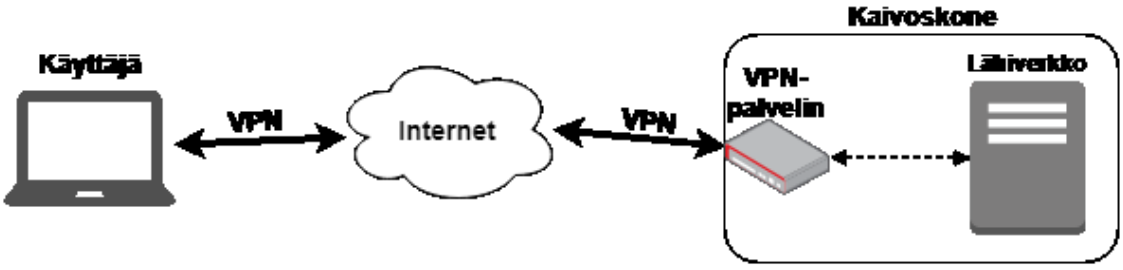
Kohdelaitteet ovat osana kaivoskoneita, joten kaivosympäristö asettaa haasteita projektille. Näytöllä saattaa pyöriä myös kaivoskoneen kriittisiä ohjausjärjestelmiä, joten etähallintaohjelman resurssien käytön tulee olla todella rajattua näissä tapauksissa. Koska kyseessä on etäkäyttöinen hallintaohjelma, tulee sinne olla joiltakin osin langaton internet-yhteys. Kaivosympäristöjen yhteydet vaihtelevat todella paljon. Parhaimmassa tapauksessa käytössä on koko kaivoksen kattava 4G-yhteys, jolla esimerkiksi VNC-protokollan pitäisi toimia erittäin sulavasti suurimman osan ajasta. Vastaavasti taas joissain on

tarjolla vain osittainen 2G-yhteys, jolla jo yksinkertaisen nettisivun palauttaminen käyttäjälle voi olla todella haastavaa yhteyden epäluotettavuuden takia.

Kaivosympäristössä mennään aina turvallisuus edellä. Tässäkin tapauksessa, kun näyttöyksikössä saattaa pyöriä kriittisiä koneenohjausjärjestelmiä, tulee järjestelmän resurssien käytön olla hallittua ja hoitua matalalla prioriteetilla. Käyttöjärjestelmä hoitaa tärkeiden prosessien skeduloinnin, joten sen suurempaa huolta ei tarvitse suorituskyvystä kantaa. Pääasiassa etähallintaohjelma pyritään kuitenkin sijoittamaan erilliseen yksikköön, jossa ei pyöri näitä kriittisiä prosesseja. Tämä myöskin helpottaa jo olemassa olevien kaivoskoneiden tuomista etähallinnan piiriin, kun vanhaa ei tarvitse muokata. Ainoastaan riittää, kun tällaiseen erilliseen laiteyksikköön kytketään kaivoskoneesta virta ja se liitetään koneen lähiverkkoon, josta etähallintaohjelma saa tarvittavat tietonsa.

## 2.4 Laiteyhteys – VPN

Puhuttaessa turvallisesta ja rajatusta yhteydestä tiettyyn verkossa sijaitsevaan laitteeseen, on VPN lähes poikkeuksetta ainoa varteenotettava vaihtoehto. VPN tarjoaa suojatun yhteyden kahden tai useamman laitteen välille ja se suojaa yksityistä verkkoliikennettä nuuskinnalta, häirinnältä ja sensuurilta (ExpressVPN 2019). VPN-yhteyttä voidaan rajoittaa helposti antamalla vain koneeseen oikeutetuille henkilöille sertifikaatit, joilla VPN-yhteyden muodostus onnistuu. Tietoturva paranee ja näin ollen koneesta ei tarvitse avata ylimääräisiä portteja palomuurista, jolloin koneet altistuisivat herkemmin hakkereiden hyökkäyksille. Kaivosympäristön välillä epäluotettavat yhteydet yhdistettynä VPN-tunneliin (kuvio 1) rajoittavat liikuteltavan datan suuruutta ja nopeutta merkittävästi.




KUVIO 1. VPN-yhteys käyttäjän ja kohdelaitteen välillä

# 3 KÄYTETYT TEKNIIKAT

Etähallintaohjelma sisältää kaksi selkeää erillistä kokonaisuutta: palvelinpuoli ja käyttöliittymä. Kun alustaksi on valittu selain, on erilaisten tekniikoiden ja ohjelmistokehysten valinnanvara lähes ääretön. Tähän projektiin valittiin käyttöliittymäkehykseksi yleisesti käytössä oleva Angular (versio 7). Angular valikoitui myös osaksi sen takia, että aiempiakin asiakkaan projekteja on toteutettu kyseisellä sovelluskehyksellä. Näin mahdollinen jatkokehitys luonnistuu helposti muidenkin henkilöiden toimesta.

Palvelinpuoli pyörii Node.js-ajoympäristössä. Alkujaan palvelinpuoli pohjautui Djangoon, mutta tämä jouduttiin vaihtamaan melkein heti projektin alussa, kun Python-ohjelmointikielen ristikäännöksessä oli ongelmia yhden näytön käyttöjärjestelmän kanssa. Node.js pohjautuu samaan JavaScript-ohjelmointikieleen kuin käyttöliittymässä käytetty Angular.

## 3.1 JavaScript-ohjelmointikieli

JavaScript on kevyt ja tulkattava ohjelmointikieli, joka on erityisesti käytössä selainapplikaatioissa ja nettisivuilla. JavaScript on prototyyppipohjainen, moniparadigmainen ja dynaaminen kieli, joka tukee oliopohjaista, imperatiivista ja deklaratiivista (esimerkiksi funktionaalinen ohjelmointi) tyyliä (Mozilla Developer Network: JavaScript 2019). Koska JavaScript pyörii pääasiassa nettisivuilla käyttäjän päätelaitteella, on sillä helppo toteuttaa erilaisia toiminnallisuuksia nettisivulle. Näitä ovat esimerkiksi erilaiset animaatiot, nappien toiminnallisuudet ja valikkojen avautuminen.

Yleinen väärinkäsitys on, että JavaScript on ”tulkattu Java”. JavaScript on täysin oma ohjelmointikieli, jonka syntaksi muistuttaa tarkoituksenmukaisesti Javaa ja C++:aa. Tämä helpottaa huomattavasti kielen oppimista vähentämällä uusia opittavia asioita. If-lausekkeet, for- ja while-silmukat sekä switch ja try-catch kokonaisuudet toimivat lähes samalla tavalla kaikissa kolmessa edellä mainitussa ohjelmointikielessä.

JavaScriptin suurimpia etuja ovat sen yksinkertaisuus ja nopeus, jos koodi ei sisällä ulkopuolisten resurssien käyttöä. Kieli on helppo oppia ja internetistä löytyy kasoittain ohjeita ja esimerkkejä erilaisten asioiden toteuttamiseen. Kielen suuri suosio on tuonut mukanaan myös haittapuolia. JavaScriptiä käytetään paljon XSS-hyökkäyksissä (Cross-Site Scripting). Siinä nettisivun tai sovelluksen JavaScript-koodi voi käyttäjän huomaamatta ladata haittaohjelmia tai muuta haitallista koodia käyttäjän koneelle. Nämä hyökkäykset ovat yleisiä sivustoilla, joissa käsitellään henkilötietoja tai maksuliikennettä.

Lisähuolta kielen kehittäjille aiheuttaa selainten vaihteleva tuki eri ominaisuuksille ja toiminnallisuuksille. Eri selaimet saattavat suorittaa saman koodipätkän eri tavalla ja tästä syystä kehittäjien tulee testata sovelluksia kaikilla selaimilla, mikä taas aiheuttaa lisäkuluja ja vie ylimääräistä aikaa. Suurin osa ominaisuuksista kuitenkin suoritetaan kaikissa selaimissa samalla tavalla ja kyseiset ongelmat liittyvät yleensä vain harvinaisiin rajatapauksiin.

### 3.1.1 ECMAScript-standardi

ECMAScript (ES) on skriptikieli, joka muodostaa pohjan JavaScriptille. ECMAScript pohjautuu ECMA-262 ja ECMA-402 standardiin, jotka molemmat ovat ECMA International -organisaation kehittämiä ja ylläpitämiä. Vuonna 2019 ECMA-262 standardista on tekeillä jo 10. versio. Uusin versio, ES10, tuo mukanaan uusia metodeja taulukoiden, objektien ja merkkijonojen käsittelyyn. Tässä projektissa käytettiin kuitenkin standardin versiota ES5, jolla varmistettiin sovelluksen toimivuus kaikissa nykyaikaisissa selaimissa.

### 3.1.2 TypeScript-ohjelmointikieli

TypeScript on tyypitetty ohjelmointikieli, joka käännetään tavalliseksi JavaScriptiksi. Se toimii kaikilla selaimilla, Node.js-ajoympäristössä tai missä tahansa JavaScript-ympäristössä, joka vain tukee ECMAScript 3 standardia tai uudempaa versiota. Kieli muistuttaa syntaksiltaan hyvin pitkälti JavaScriptiä, joten kielen opettelu JavaScriptin jo osaavalle ei ole vaikeaa. TypeScript asennetaan projektiin helpoiten npm-paketinhallinnan avulla. Monet koodieditorit

sisältävät jo itsessään tarvittavat työkalut TypeScript-kielen kirjoittamiseen ja tarkistamiseen.

Typescript korjaa JavaScriptin suurimpia puutteita ja sudenkuoppia. Esimerkiksi JavaScriptissä voidaan merkkijono-muuttujaan missä vain vaiheessa syöttää tavallinen kokonaislukuarvo (kuva 2). Tässä tapauksessa toisen henkilön kirjoittamaa koodia lukevan on välillä vaikea tietää muuttujan sisältöä ja tyyppiä, jos koodi ei ole huolellisesti ja selkeästi kirjoitettua. TypeScript mahdollistaa muuttujan tyypityksen merkkijonoksi, jolloin viimeistään käännösvaiheessa kääntäjä ilmoittaa virheellisestä tyyppimuunnoksesta (kuva 3), mutta useimmat koodieditorit ilmoittavat virheestä jo kirjoitusvaiheessa. Alla olevat esimerkit ovat Visual Studio Code -editorista ja TypeScript esimerkki on projektista, jossa kyseinen kieli virhetarkasteluineen on käytössä.

```
3   var title = "Otsikko";
4   console.log(title); // Tuloste: Otsikko
5   title = 2019;
6   console.log(title); // Tuloste: 2019
```

KUVA 2. JavaScript mahdollistaa muuttujan tyypin vaihtamisen

```
15    let title: string = 'Otsikko';
16    console.log(title); // Tuloste: Otsikko
17    title = 2019; // Type '2019' is not assignable to type 'string'.
```

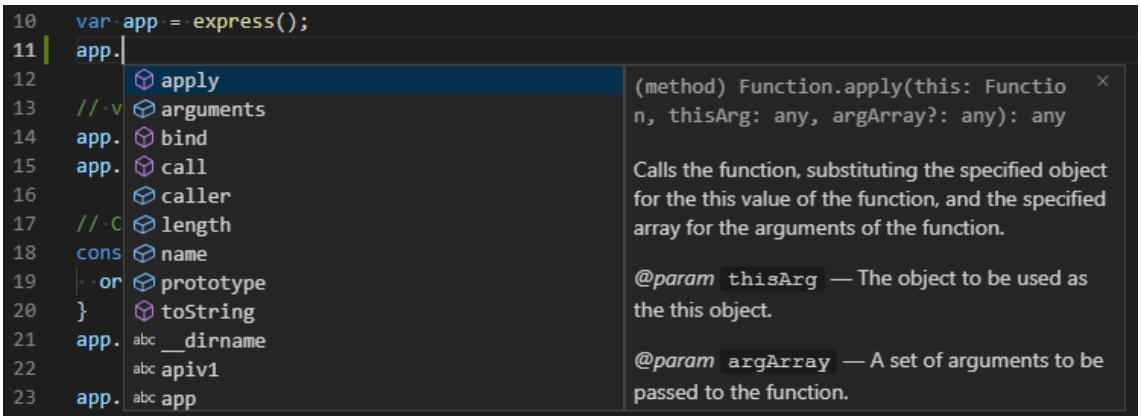
KUVA 3. TypeScript antaa virheen muuttujan eroavasta tyypistä

### 3.2 Kehitysympäristö

Kehitysympäristö kattaa tässä projektissa tietokoneen ja yhden Epec 6107 mallisen näyttölaitteen. Varsinainen kehitys tapahtuu virtuaalikoneessa, jossa pyörii Xubuntu 18. Kyseessä on Linux-pohjainen käyttöjärjestelmä, joka helpottaa erilaisten skriptien (liite 1) ja työkalujen käyttöä, kun kohdelaitteidenkin käyttöjärjestelmät pohjautuvat Linux-ytimeen. Teknisten dokumentaatioiden kirjoittaminen ja web-käyttöliittymän testaus kuitenkin tapahtuvat virtuaalikoneen ulkopuolella tarvittavien työkalujen ja resurssien vuoksi.

### 3.2.1 Visual Studio Code -tekstieditori

Visual Studio Code on moderni, kevyt ja tehokas tekstieditori, joka on saatavilla Windowsillle, macOS:lle ja Linuxille. Siihen sisältyy sisäänrakennettu tuki JavaScriptille, TypeScriptille ja Node.js:lle. Tämän lisäksi editorin ympärillä on todella laaja ekosysteemi, josta löytyy lisäosia ja laajennuksia useille kielille kuten C++, C#, Java, Python, PHP ja Go sekä ajoympäristöt esimerkiksi .NET:lle ja Unitylle. Näiden lisäksi ehdottomasti paras ominaisuus on editorin mukana tuleva IntelliSense (kuva 4). IntelliSense on Microsoftin kehittämä ohjelma ja termi erilaisille koodinkäsittelyn mahdollisuuksille, joita ovat esimerkiksi koodin täydennys, parametri-info, funktion tarkastelu ja muuttujalistaus. Erityisesti koodin täydennys ja muuttujalistaus ovat hyödyllisiä ja koodaamista nopeuttavia asioita, kun koko sanaa ei tarvitse kirjoittaa eikä muuttujien nimiä muistaa täysin ulkoa. Samalla vältytään myös kirjoitusvirheiltä, joiden korjaaminen saattaa joskus olla todella hankalaa ja aikaa vievää.

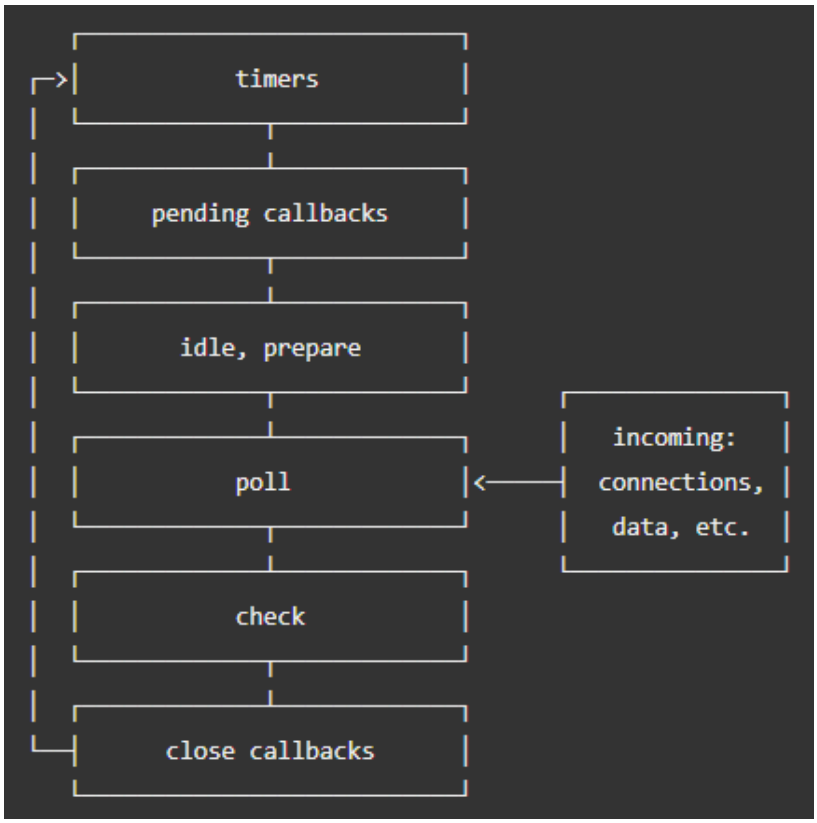



KUVA 4. Visual Studio Code IntelliSense

### 3.2.2 Node.js-ajoympäristö

Node.js on asynkroninen ja tapahtumapohjainen JavaScript-ajoympäristö, joka on rakennettu Googlen kehittämän avoimen lähdekoodin V8 JavaScript moottorin päälle. Node.js sopii erityisesti skaalautuvien web-applikaatioiden luomiseen. Mutkattoman skaalautumisen mahdollistaa lukkojen puuttuminen. Säie-pohjaiset prosessit ovat huonoja web-applikaatioissa, koska ne ovat vaikeita käyttää ja hyvinkin tehtyinä tehottomia. Node.js ei käytä lainkaan lukkoja vaan tapahtumat

pyörivät tapahtumasilmukassa (englanniksi event loop, kuvio 2). Silmukassa käydään läpi erilaisia vaiheita, joista yksi on näiden tapahtumien suorittaminen. Jokaisen kierroksen aikana suoritetaan n-määrä tapahtumia tai toisin sanoen takaisinkutsuja. Takaisinkutsuja käsitellään yhden kierroksen aikana joko niin kauan, että ne loppuvat tai kutsujen maksimimäärä tulee vastaan. Tämän jälkeen silmukka siirtyy seuraavaan vaiheeseen ja jatkaa kutsujen suorittamista taas seuraavalla kierroksella. Valmistuneet takaisinkutsut suljetaan silmukan kierroksen lopussa ja tämän jälkeen ne eivät vie enää aikaa seuraavalla kierroksella.




KUVIO 2. Node.js tapahtumasilmukka

### 3.2.3 Lighttpd-palvelinohjelma

Lighttpd on avoimen lähdekoodin HTTP-palvelinohjelma, jota käytetään etähallintatyökalun käyttöliittymän WWW-palvelimena. Ohjelma on suunniteltu turvalliseksi, nopeaksi ja joustavaksi (Lighttpd 2019). Tällöin se soveltuu erinomaisesti kyseisen projektin kohdelaitteisiin, joissa saatavilla olevat resurssit ovat rajalliset. Ohjelma on monipuolisesti mukautettavissa asetustiedostojen

avulla ja dokumentaatio tähän on kattava ja selkeä. Angularin vaatiman virheellisten reittien uudelleenohjauksen määritys onnistui sekin vaivattomasti.

### 3.2.4 Npm-ohjelmistorekisteri

Npm eli Node Package Manager on maailman suurin ohjelmistorekisteri (npm 2019). Avoimen lähdekoodin kehittäjät käyttävät sitä pakettien ja lisäosien lataamiseen ja jakamiseen. Järjestelmä koostuu nettisivusta, komentorivityökalusta (englanniksi Command Line Interface, CLI) ja rekisteristä. Nettisivulta voi tarkastella rekisterin paketteja ja niiden tietoja kuten esimerkiksi dokumentaatiota, versionumeroa, lisenssiä tai avointen ongelmien ja latausten määrää. Komentorivityökalulla ladataan halutut paketit rekisteristä omalle tietokoneelle tai toisinpäin.

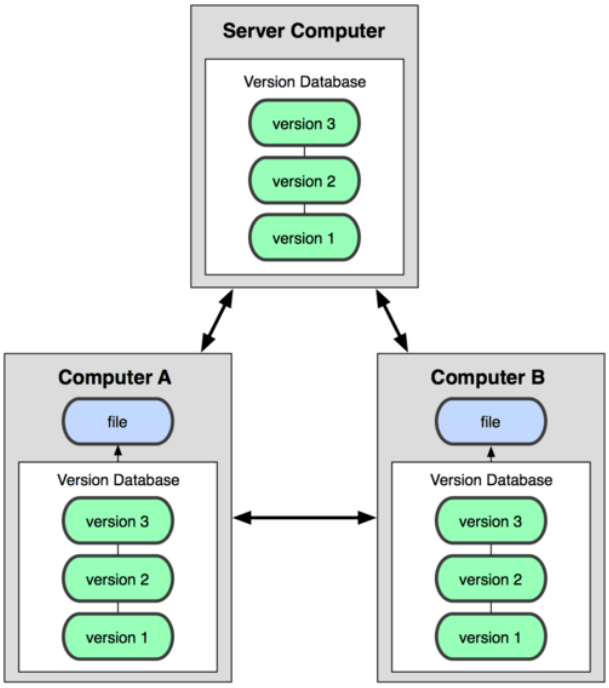
## 3.3 Versionhallinta

Versionhallinta on järjestelmä, joka ajan kuluessa tallentaa muutoksia tiedostoon tai joukkoon tiedostoja. Eri ajankohtien tiedostoversioihin on helppo palata, joten ongelmien syntyessä on joskus helpompaa vain palauttaa aiempi toimivaksi todettu versio ja yrittää uudelleen. Versionhallintajärjestelmiä on kolme erilaista: paikallinen, keskitetty ja hajautettu.

Paikallinen versionhallinta on yleinen ja yksinkertainen tapa versioiden ylläpitämiseen. Yleinen esimerkki tästä ja monien ihmisten käyttämä tapa on vain kopioida tiedostot toiseen kansioon. Nokkelimmat saattavat jopa lisätä aikaleiman kansioiden nimeen, jotta tiedostot on edes jotenkin mahdollista pistää aikajärjestykseen jälkikäteen. Koska kyseinen prosessi on monesti täysin manuaalinen, on sen käyttö hankalaa, hidasta ja riskialtista. Helposti unohdetaan missä mikäkin tiedosto on, mitä kyseinen versio sisältää ja mitä muutoksia siinä on vaikka aiempaan versioon verrattuna. Tai sitten ihan vain epähuomiossa kopioidaan tiedostot edellisten päälle, jolloin aiempi versio on tietyissä ympäristöissä saman tien menetetty. Tämä tapa on suositeltava ainoastaan, jos kyseessä on vain yksi tai muutama tiedosto, joista tiedetään jo etukäteen, että uusia versioita tulee mahdollisesti vain muutaman kerran.

Keskitetty versionhallintajärjestelmä on jo askel parempaan suuntaan. Siinä versionhallintajärjestelmä on erillisellä palvelimella käyttäjän oman tietokoneen sijasta. Tämä mahdollistaa sen, että useampi ihminen voi työskennellä saman projektin tai samojen tiedostojen parissa yhtä aikaa. Järjestelmä myöskin mahdollistaa kontrollin versionhallinnan käyttäjille, kun ylläpitäjä voi antaa kullekin juuri ne oikeudet, joita käyttäjä oikeasti tarvitsee. Näin estetään myös helposti kokemattomampien käyttäjien mahdolliset virheelliset komennot, jotka pahimmassa tapauksessa voisivat sotkea koko projektin versiohistorian. Haittapuolena tulee kuitenkin järjestelmän yksi ainoa palvelin, joka rikkoutuessaan voi kerralla viedä koko versiohistorian mennessään, jos kunnollisia varmuuskopioita ei ole olemassa tai jos ne ovat todella vanhentuneet. Tunnetuin keskitetty versionhallintajärjestelmä on Subversion.

Keskitetyn versionhallintajärjestelmän puutteita korjaamaan on kehitetty hajautetut versionhallintajärjestelmät. Näissä käyttäjät eivät hae tiedostojen viimeisintä versiota vaan sen sijaan kopioivat itselleen täydellisen historian kaikista tiedostoista (kuvio 3). Näin ollen palvelimen tuhoutuessa on erittäin suuri todennäköisyys, että yhdeltä tai useammalta käyttäjältä löytyy lähes viimeisin ja täydellinen historia, joka voidaan helposti palauttaa uudelle palvelimelle. Tunnetuimpia hajautettuja versionhallintajärjestelmiä ovat Git ja Mercurial.




KUVIO 3. Hajautettu versionhallintajärjestelmä (Git 2019)

### 3.3.1 Git-versionhallinta

Git on vuonna 2005 alkunsa saanut Linus Torvaldsin kehittelemä hajautettu versionhallintajärjestelmä. Torvalds tunnetaan myös Linux ytimen alkuperäisenä kehittäjänä. Idea Gitin rakentamiseen syntyi, kun Linux kernel -projektin kehittäjäyhteisö kaipasi kipeästi uutta versionhallintajärjestelmää suhteiden katkettua edellistä työkalua kehittävän yhtiön kanssa. Syntymästään lähtien Git on kehittynyt monien mielestä helpoksi käyttää ja siitä huolimatta se on säilyttänyt alkuperäiset ominaisuutensa, nopeutensa ja tehokkuutensa suurienkin projektien kanssa. Gitin suurimpana kilpailijana on pidetty Subversion-järjestelmää, joka tosin on viime vuosina jäänyt käyttäjämäärissä paljon Gitin jalkoihin.

### 3.3.2 Gerrit Code Review -katselmointityökalu

Gerrit tuo versionhallintaan oman lisänsä mahdollistamalla koodimuutosten katselmoinnit. Järjestelmä on alkujaan lähtöisin Gerrit Rietveldin kehittämästä koodikatselmointityökalusta, jota sittemmin on muokattu melko rajusti. Gerrit tukee useampia versionhallintajärjestelmiä ja se tulee eräänlaiseksi solmuksi käyttäjän ja palvelimen väliin. Käyttäjä julkaisee muutoksensa versionhallintaan, josta ne siirtyvät Gerritiin katselmoitavaksi. Vielä tässä vaiheessa koodi ei siis ole kaikkien nähtävillä julkisessa versionhallinnassa. Katselmointivaiheessa muutokselle valitaan yksi tai useampi katselmoija, jotka voivat kommentoida ja pisteyttää muutosta sen oikeellisuuden ja tarpeellisuuden mukaan. Vasta kun muutos on saanut tietyn määrän pisteitä ja mahdollisia korjauksia puutteisiin, voidaan se hyväksyä ja liittää osaksi julkista versionhallintaa.

Koodin katselmointi vähentää, mutta ei kuitenkaan poista rikkinäisen koodin pääsemistä kehitettävään projektiin. Kun toinen ihminen katsoo koodia ja toivottavasti myös lukee ajatuksella, on hänen helppo kyseenalaistaa ja huomata mahdolliset epäkohdat, joita koodi saattaa sisältää. Tässä vaiheessa niihin on vielä nopeaa puuttua ja korjaavia muutoksia on helppo tehdä ennen lopullisen version julkaisua osaksi julkista versionhallintaa.

Monesti projektin takarajat määräävät, miten tarkasti ja millä panoksella koodia katselmoidaan. Demo-tarkoitukseen tuotettavan sovelluksen katselmointi voi olla

todella vähäistä, jos sitä on edes lainkaan. Vastaavasti ison ja kriittisen järjestelmän katselmointiin voidaan käyttää jopa saman verran aikaa kuin itse koodin kirjoittamiseenkin. Tässä projektissa katselmointia suoritettiin vain kriittisimpien kohtien osalta.

### 3.4 Ohjelmistokehykset

Ohjelmistokehykset, tai toiselta nimeltään sovelluskehykset, helpottavat ja nopeuttavat ohjelmistojen tuottamista ja kehittämistä. Ajatusmalli niiden pohjalla on ikään kuin ”miksi tehdä jotain, minkä joku toinen on jo tehnyt”. Sovelluskehyksellä voidaan tyypistä riippuen lisätä nopeasti pieniä tai suurempia kokonaisuuksia omaan kehitettävään ohjelmistoon. Yksi esimerkki tällaisesta on esimerkiksi kalenterin tuominen omalle nettisivulle. Internet on pullollaan JavaScriptillä tehtyjä kalenterikirjastoja. Niistä tarvitsee vain napata mieleinen ja mahdollisesti myös tunnettu kirjasto ongelmien välttämiseksi. Avoimen lähdekoodin kirjasto on myös iso plussa, sillä lisenssistä riippuen se mahdollistaa kyseisen kirjaston räätälöinnin omiin tarkoituksiin sopivaksi. Avoimesta lähdekoodista on helpompi varmistua, ettei kirjasto tai sovelluskehys tee mitään ylimääräistä, kuten esimerkiksi lähetä joitain tärkeitä tietoja jonnekin etäiselle palvelimelle.

Tässä projektissa erilaisia ohjelmistokehyksiä on käytössä lukuisia ja kaikkien esitteleminen ei ole tarpeellista. Seuraavissa alaotsikoissa käsitellään vain projektissa näkyvimmässä roolissa olevat ja tarpeellisimmat ohjelmistokehykset.

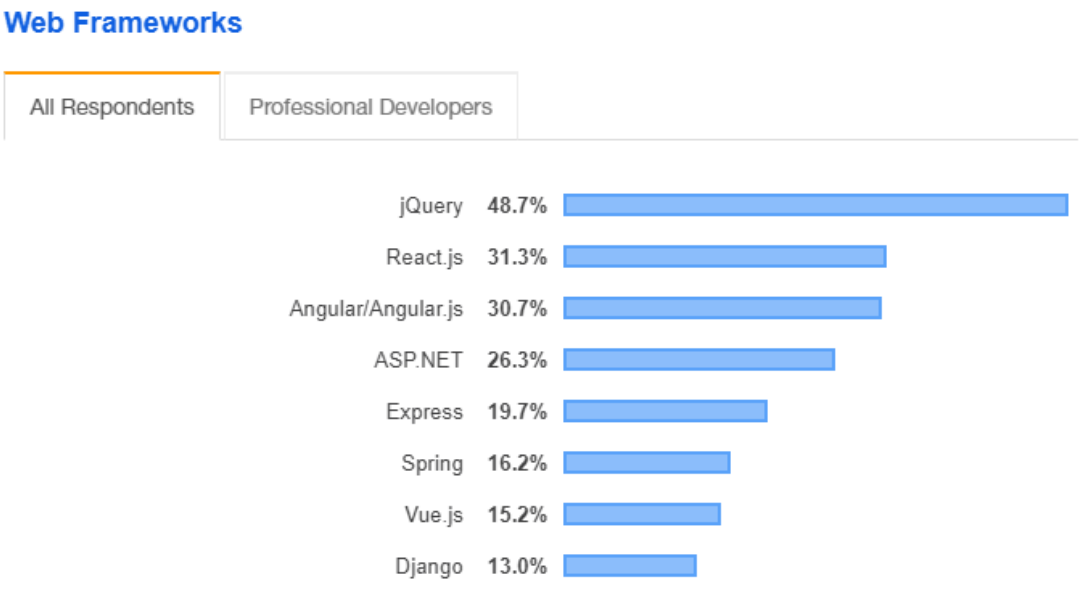
#### 3.4.1 Angular-ohjelmistokehys

Angular on nopea ja suorituskykyinen ohjelmistokehys web-sovelluksien kehitykseen. Sillä on mahdollista luoda monimutkaisia ja suuriakin web-sovelluksia, jotka pyörivät kaikilla alustoilla niin mobiilissa kuin työpöytäjärjestelmissäkin. Web-sovellusten suurimpiin etuihin kuuluu yhteensopivuuden lisäksi asennuksien puute. Käyttäjä tarvitsee vain yleiskäyttöisen modernin selaimen ja kaikki on valmista sovelluksen käyttöön. Angularin kehitystä varten tarvitsee osata vain HTML, CSS, JavaScript sekä

MVC-arkkitehtuurimallin (englanniksi Model-View-Controller) perusidea. Vaikkakin Angular käyttää ohjelmistologiikan kielenä TypeScriptiä, muistuttaa se hyvin pitkälti JavaScriptiä, jolloin TypeScriptin opettelu ei tule ongelmaksi. Loppujen lopuksi TypeScript käännetään natiiviksi JavaScriptiksi, kun sovelluksesta käännetään staattinen versio.

Viimeisen parin vuoden aikana Angular ja sen mahdollisesti pahin kilpailija React.js (myöh. React) ovat olleet pinnalla web-sovellusten ohjelmistokehyksistä keskusteltaessa. Molempien kehysten takana ovat suuret tietojätit, Angularilla Google ja Reactilla Facebook. Stack Overflow:n kyselyn (kuvio 4) mukaan nämä kaksi kehystä hallitsevat kärkisijoja ja edelle mahtuu ainoastaan jQuery. Näiden kahden kirjaston suurimpana erona lienee oppimiskäyrä. Angularissa se on suurempi, sillä kehys pitää sisällään lähes kaiken tarvittavan suurenkin web-sovelluksen kehittämiseen. React taas noudattaa hieman modulaarisempaa ajatusmallia. Siinä peruspaketissa tulee ainoastaan kaikki pakollinen ja muun tarvittavan kehittäjä lisää itse tarpeidensa mukaan.

Stack Overflow:n vuonna 2019 järjestämän kehittäjäkyselyn perusteella React.js oli noin 0,6 % suositumpi Angulariin verrattuna. Tilanne kääntyy kuitenkin Angularin eduksi 0,1 % turvin (Angular/Angular.js 32,4 %, React.js 32,3 %), kun tuloksiin lasketaan vain ammattilaisten vastaukset. Kyselyyn vastasi 63 585 henkilöä, joista 55 079 oli ammattilaisia. (Stack Overflow 2019)




KUVIO 4. Web-ohjelmistokehysten suosio (Stack Overflow 2019)

Angularista julkaistaan uusi pääversio aina puolen vuoden välein, pääversion korjauksia ja muutoksia 1-3 kappaletta ja pieniä korjauksia lähes joka viikko (Angular 2019). Tiheä julkaisuaikataulu mahdollistaa ongelmien korjaamisen ja muutosten julkaisemisen lyhyellä aikataululla sen sijaan, että pienen ongelman korjaaminen näkyisi käyttäjille vasta mahdollisesti jopa useiden vuosien päästä. Tämän projektin aloitushetkellä Angularin viimeisin julkaisuversio oli Angular 6. Projektin edetessä kehyksestä julkaistiin uusi pääversio, johon projekti myöhemmin päivitettiin. Angular asennetaan npm-komentorivityökalun avulla. Projekti alustetaan Angularin oman komentorivityökalun avulla, jossa on mahdollista valita käytetty tyyliohje ja tarvittavat lisäominaisuudet.

### 3.4.2 RxJS-ohjelmistokehys

RxJS on kirjasto, joka mahdollistaa reaktiivisen ohjelmoinnin käyttäen ”kuunneltavia” (englanniksi observables). Kuunneltavat helpottavat asynkronisen koodin ja takaisinkutsujen kirjoittamista. Yhtä kuunneltavaa kohden voidaan lisätä n-määrä kuuntelijoita, jotka reagoivat kuunneltavan lähettämään dataan halutulla tavalla (koodiesimerkki, kuva 5). RxJS on tärkeä osa Angularia ja sitä käytetään muun muassa arvojen iterointiin tietovirrasta, kartoittamaan arvoja ja muuttamaan niiden tyyppiä, tietovirtojen suodatukseen ja yhdistämään useita tietovirtoja.

Create an observable from a promise

```
import { from } from 'rxjs';

// Create an Observable out of a promise
const data = from(fetch('/api/endpoint'));
// Subscribe to begin listening for async result
data.subscribe({
  next(response) { console.log(response); },
  error(err) { console.error('Error: ' + err); },
  complete() { console.log('Completed'); }
});
```

KUVA 5. Kuunneltavan luominen API-kutsusta (Angular 2019)

Edellisessä kuvassa kuunneltava luodaan HTTP-protokollan API-kutsusta, jossa haetaan dataa palvelimen päätepisteestä. Päätepisteestä saatu data syötetään tietovirtana objektille, josta tässä tapauksessa luodaan samalla kuunneltava. Kuunneltavan tietovirtaa aletaan kuuntelemaan ”subscribe”-metodilla. Metodiin voidaan antaa kolme omaa metodia eri tilanteiden varalle. Seuraava, eli ”next”-tilanne suoritetaan, kun kuunneltavan data vastaanotetaan onnistuneesti. Vastaavasti suoritetaan ”error”-tilanne, jos kuunneltavan data sisältää virheen. Onnistuneessa tapauksessa suoritetaan myös ”complete”-tilanne, kun tietovirta on kokonaan vastaanotettu.

### 3.4.3 Express.js-ohjelmistokehys

Express.js on todella suosittu web-sovelluskehys palvelinpuolten ja osin myös käyttöliittymien luomiseen. Se on kevyt ja joustava sovelluskehys Node.js-ajoympäristöön. Express.js sisältää lukemattomia metodeja ja väliohjelmistoja (englanniksi middleware), jotka mahdollistavat turvallisen ja toimivan API:n tekemisen helposti ja nopeasti. Express.js asennetaan npm-komentorivityökalun avulla. Alla olevasta esimerkistä (kuva 6) käy nopeasti ilmi kehyksen käytön yksinkertaisuus ja selkeys.

```
var express = require('express')
var app = express()

// respond with "hello world" when a GET request is made to the homepage
app.get('/', function (req, res) {
  res.send('hello world')
})
```

KUVA 6. Yksinkertainen polkuesimerkki

### 3.4.4 JSON Web Tokens -ohjelmistokehys

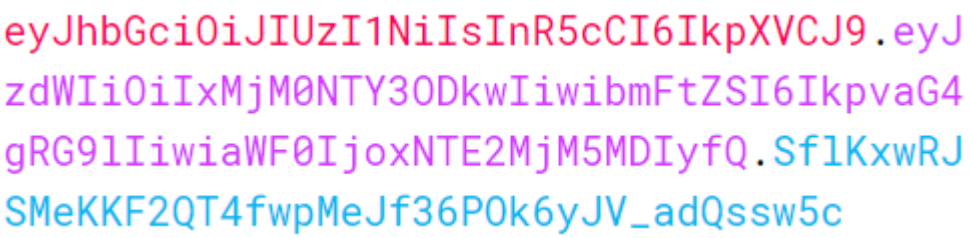
JSON Web Tokens on avoimen lähdekoodin ja standardin RFC 7519 mukainen ohjelmistokehys käyttöoikeustietueiden välittämiseen kahden osapuolen välillä (JSON Web Tokens 2019). Kehyksellä on helppo luoda, allekirjoittaa ja

varmentaa tietueita, joiden avulla voidaan välittää palvelinpuolelta käyttäjälle kirjautumistiedon perusteella käyttäjän avain. Käyttäjän avain lähetetään palvelinpuolelle jokaisen seuraavan kutsun yhteydessä ja palvelin varmentaa avaimen. Jos avaimen varmennus onnistuu, palautetaan käyttäjälle tämän pyytämät resurssit. Muussa tapauksessa estetään tietojen palauttaminen ja annetaan käyttäjälle virheilmoitus ja mahdollisuus varmentaa itsensä uudestaan esimerkiksi kirjautumalla sisään.

JSON Web Token koostuu kolmesta osasta, jotka on erotettu toisistaan pisteellä. Nämä kolme osaa ovat

- ylätunniste
- hyötykuorma
- allekirjoitus.

Alla olevassa esimerkissä (kuva 7) ensimmäinen osio (punainen teksti) esittää ylätunnistetta, joka pitää sisällään käytetyn algoritmin ja tietuetyypin. Seuraava osio (violetti teksti) esittää hyötykuormaa, jonka sisältö on vapaamuotoinen. Viimeinen osio (vaaleansininen teksti) esittää allekirjoitusta. Tietue tallennetaan yleensä selaimen kekseihin, paikalliseen muistiin tai jonnekin sovelluksen sisäiseen muistipaikkaan. Tietue voidaan lähettää usealla tavalla palvelinpuolelle pyynnön yhteydessä. Tapoja käsitellään tarkemmin kappaleessa 5.3.




KUVA 7. Allekirjoitettu tietue (JSON Web Token 2019)

### 3.5 Jatkuva integraatio

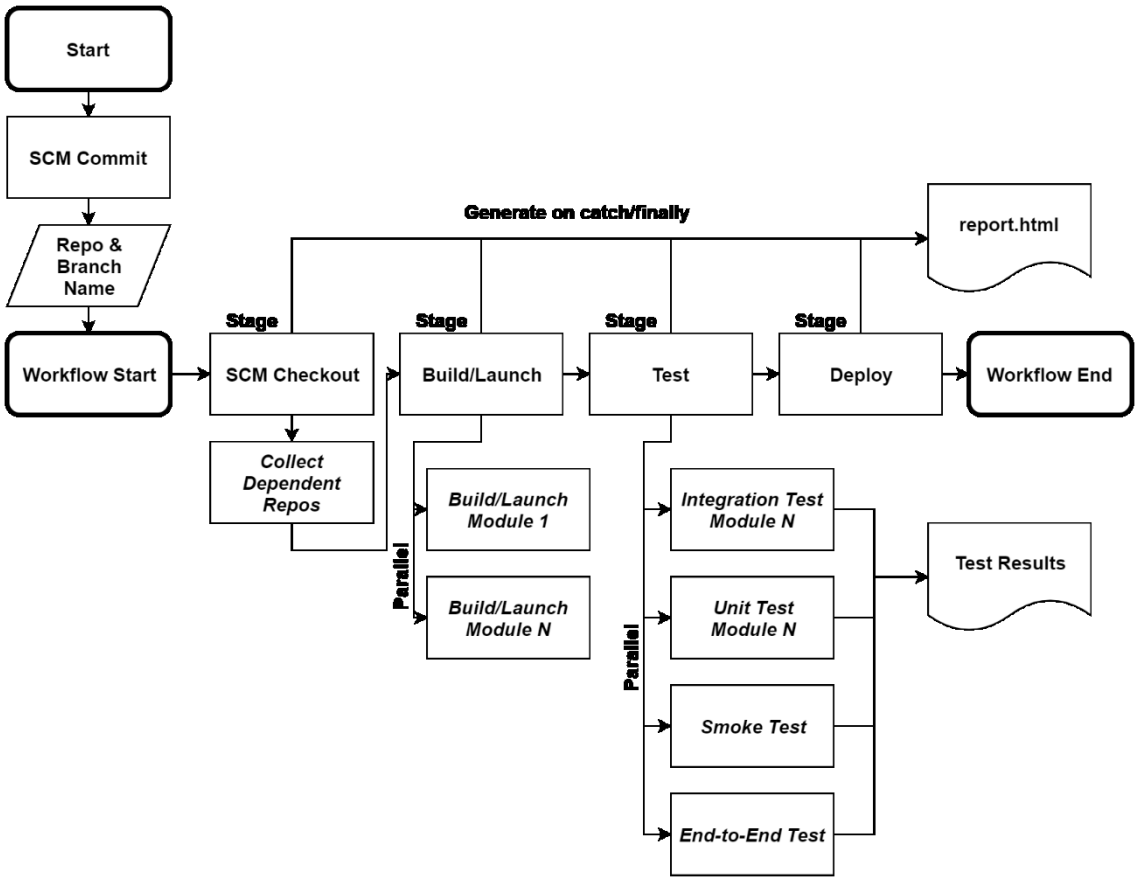
Jatkuvalla integraatiolla tarkoitetaan ohjelmistotuotannon alalla hyväksi havaittuja automaattisia prosesseja. Lähdekoodin tai projektin muutokset yhdeltä

tai useammalta tekijältä kootaan automaattisesti versionhallinnasta, ja tämän jälkeen niille suoritetaan tiettyjä ennalta määritettyjä vaiheita. Vaiheet voivat vaihdella aina yksikkötestien suorittamisesta muutoslokin automaattiseen päivittämiseen versionhallinnan pohjalta. Tämän lisäksi voidaan projektit kääntää haluamille alustoille, jolloin esimerkiksi erilaisia asennusmedioita ei kehittäjien tarvitse kääntää ja luoda käsin. Automaatiojärjestelmät ovat usein todella skaalautuvia, joten suurienkaan projektien ja tiheiden käännösaikavälien suorittaminen ei muodostu ongelmaksi, kunhan vain resursseja on tarpeeksi käytössä.

Automatisoituja integraatioprosesseja voidaan käyttää helposti myös koodikatselmoinnin yhteydessä. Järjestelmä saa ilmoituksen uudesta odottavasta koodikatselmoinnista ja alkaa tämän jälkeen suorittamaan sille ennalta määritettyjä tehtäviä. Näitä tehtäviä voivat esimerkiksi olla yksikkötestien ajaminen ja jonkin yleisen käännöksen suorittaminen, jotta voidaan todeta kaiken olevan kunnossa. Jos yksikkötesteissä ei ole virheitä ja käännös onnistuu, voi järjestelmä automaattisesti antaa koodikatselmoinnille yhden lisäpisteen. Tämänkin jälkeen ihmisen on vielä hyvä katselmoida koodi, sillä kaikki ongelmakohdat eivät välttämättä näy vielä automaattisissa testeissä.

### 3.5.1 Jenkins-automaatiojärjestelmä

Jenkins on eräs tunnetuimmista jatkuvan integraation järjestelmistä ja sen lähdekoodi on täysin avoin. Se on todella monipuolinen ja skaalautuva järjestelmä, joka voidaan helposti laajentaa jatkuvan integraation piiristä myös jatkuvan jakelun puolelle. Sen todella laajan lisäosavalikoiman ansiosta jokainen käyttäjä pystyy räätälöimään järjestelmänsä täysin itselleen sopivaksi. Järjestelmän toimintoja ja projektien suoritettavia putkistoja (kuvio 5) ohjataan helppokäyttöisen ja selkeän web-käyttöliittymän kautta.



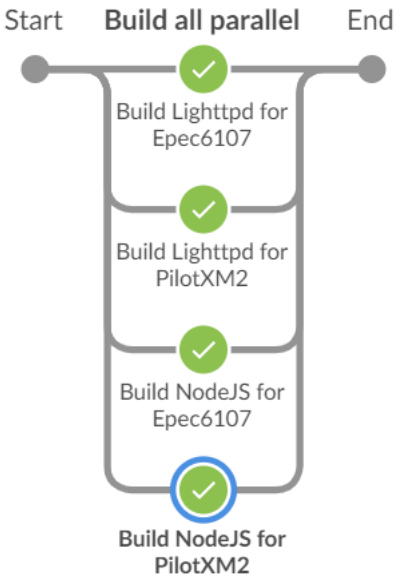

KUVIO 5. Jenkins-putkiston arkkitehtuuriesimerkki (Jenkins 2019)

Jenkinsin jokainen projekti koostuu joko yksittäisestä suoritettavasta tehtävästä tai useasta tehtävästä, jolloin niistä käytetään nimitystä putkisto. Putkisto voi sisältää n-määrän tehtäviä ja jokainen tehtävä suoritetaan toinen toisensa jälkeen tai rinnakkain riippuen siitä, miten putkisto on määritelty. Putkistojen ajaminen voi kestää muutamasta sekunnista useisiin tunteihin tai jopa päiviin riippuen tehtävien vaativuudesta ja niihin kuluvasta ajasta. Jenkins ajaa tehtäviä aina suorittajilla, jotka ovat yleensä erillisiä tietokoneita: fyysisiä tai virtualisoituja. Pääjärjestelmä jakaa suoritettavat tehtävät automaattisesti aina vapaiden suorittajien kesken tai laittaa tehtävät jonoon odottamaan, jos suorittajia ei ole sillä hetkellä vapaana.

### 3.5.2 Projektin automaattinen julkaisuprosessi

Tässä projektissa luotiin kaksi putkistoa suorittamaan tarvittavat tehtävät. Ensimmäinen putkisto (kuva 8) pitää huolen tarvittavien ajoympäristöjen

kääntämisestä kaikille kohdelaitteille. Käännettäviä ajoympäristöjä on kaksi, Lighttpd ja Node.js. Kohdelaitteita on myös kaksi, joten käännöstehtäviä on yhteensä neljä. Kaikilta Jenkinsiin määritellyiltä suorittajilta löytyy tarvittavat työkalut ympäristöjen ristikäännöstä varten jokaiselle kohdelaitteelle.




KUVA 8. Jenkins putkisto, jossa tehtävät ajetaan rinnakkain

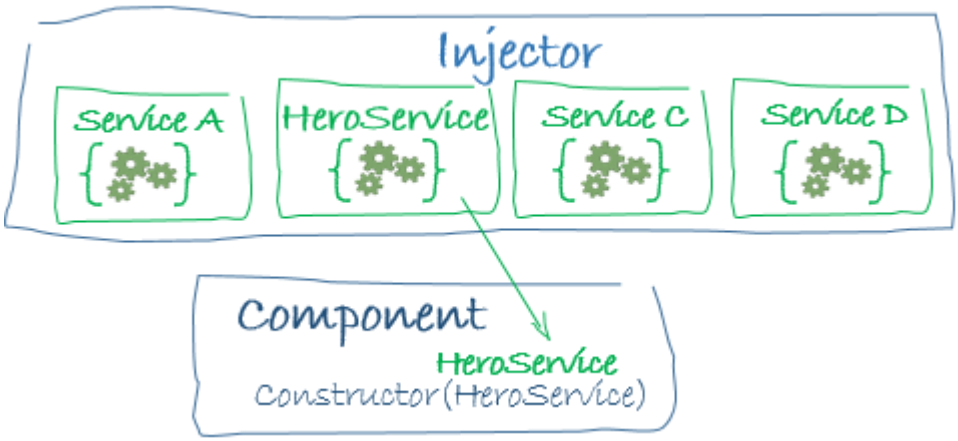
Toinen putkisto pitää huolen projektin kääntämisestä. Koska projektin lopputuloksena syntyy käyttöliittymän osalta vain staattinen web-sovellus, jota ajetaan Lighttpd-ajoympäristössä, ei käyttöliittymästä tarvitse kääntää kuin yksi versio. Sama pätee myös palvelinpuoleen, jonka JavaScript-sovellusta ajetaan Node.js-ajoympäristössä. Itse projektin kääntämiseksi asennuspakettiin pitää kuitenkin lisätä jokaisen kohdelaitteen omat skriptit ja mahdolliset lisätiedostot, joten niitä varten on tehty pienet tehtävät jokaiselle kohdelaitteelle.

Putkistojen viimeisessä vaiheessa käännösten tuloksena syntyneet tiedostot ja paketit ladataan asiakkaan pilvipalvelimelle niiden käyttöönottoa varten. Ajoympäristöt löytyvät omasta polustaan ja etähallintatyökalun asennusmediat sekä muut tiedostot omastaan. Kaikki tiedostot on versionumeroitu ja mahdolliset versionhallinnan haarat menevät omiin nimettyihin polkuihinsa sekaannusten välttämiseksi. Pilvipalvelimelle etähallintatyökalun pääkansioon luodaan käännösten yhteydessä myös muutosloki versionhallinnan muutoslokin pohjalta.

# 4 KÄYTTÖLIITTYMÄ

Käyttöliittymän suunnittelussa huomioitiin yksinkertaisuus ja toimivuus. Selainpohjainen käyttöliittymä antaa useita mahdollisuuksia sekä tuo mukanaan joitakin rajoituksia. Pääasiassa käyttöliittymän ominaisuudet ovat kuitenkin kevyitä, eivätkä vaadi mitään erityistä optimointia sen suhteen. Graafinen toteutus sekä värimaailma sovitettiin asiakkaan muiden tuotteiden väri- ja suunnittelumaailmaan sopiviksi. Käyttöliittymän kieli on ensimmäisessä versiossa englanti, mutta kielituki muille kielille on mahdollista jatkokehitysvaiheessa.

Käyttöliittymän tarvitsemat tiedot ladataan palvelinpuolelta sovelluksen aukaisemisen yhteydessä ja tallennetaan sovelluksen muistiin erinäisiin palveluihin (englanniksi services). Angularissa palveluita voidaan lisätä minkä tahansa komponentin käyttöön, joten saman tiedon jakaminen ja käyttäminen useasta komponentista on todella helppoa. Palveluiden sisältämä tieto injektoidaan komponenttiin (kuvio 6) Angularin suuttimen kautta (englanniksi injector).

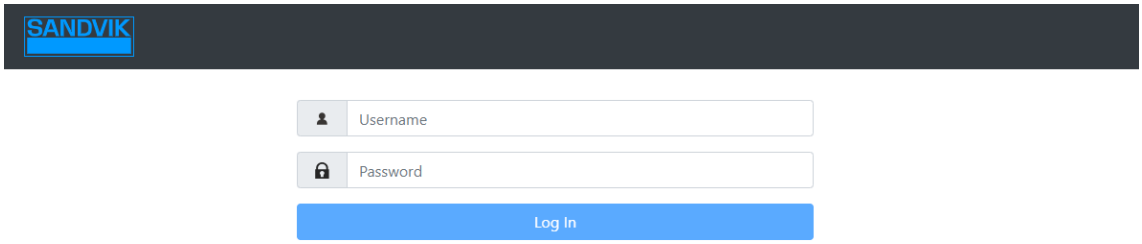



KUVIO 6. Palvelun injektointi komponenttiin (Angular 2019)

## 4.1 Vaatimukset

Käyttäjäryhmä koostuu pääasiassa asiakkaan insinööreistä ja asiantuntijoista, joten sen tarkempia selityksiä tai ohjenappeja ei tarvitse tehdä.

Sisäänkirjautuminen (kuva 9) on kuitenkin pakollista ennen applikaation avaamista, sillä jatkokehitysvaiheessa asiakkaan insinööritkin saavat omat käyttäjätasonsa ja niitä vastaavat toiminnallisuudet. Ensimmäisessä versiossa riittää kuitenkin yksi käyttäjä ja salasana, joka on muokattavissa asetustiedostossa.




KUVA 9. Käyttöliittymän kirjautumissivu

Muut vaatimukset käyttöliittymällä olivat kohtalaisen vapaat. Järjestelmän hetkittäinen saatavuuskatko ei vielä aiheuta mitään vakavaa, joten saatavuuden takaaminen ei vaadi suurempia toimenpiteitä. Liittymästä ei hoideta mitään järjestelmäkriittisiä toimenpiteitä eikä se ole kytköksissä mihinkään vastaavaan. Käyttöliittymän lopputuloksena syntyy staattinen HTML-sivusto, jota pyörittää erillinen Lighttpd-ajoympäristö. Ajoympäristönä Lighttpd on todella kevyt ja sopii erinomaisesti pienten nettisivujen ja web-sovellusten pyörittämiseen.

Käyttöliittymän responsiivisuus huomioitiin jo suunnitteluvaiheessa. Tällä hetkellä sovellusta käytetään pääasiassa kiinteiltä tai kannettavilta työasemilta, joissa on yleensä suurehko ja teräväpiirtoinen näyttö. Tulevaisuudessa on kuitenkin mahdollista, että sovellusta tullaan käyttämään myös kannettavilta mobiililaitteilta. Käyttöliittymä skaalautuu erikokoisille näyttöpäätteille automaattisesti ja erilaisten elementtien koko ja sijoittelu on suunniteltu niin, että sovellusta on helppo ja selkeä käyttää kaiken kokoisilta laitteilta. Käyttöliittymän ulkoasun rakentamisessa käytettiin apuna erittäin suosittua ja helppokäyttöistä Bootstrap 4 -kirjastoa. Kirjasto mahdollistaa elementtien responsiivisen sijoittelun sen ruudukkomaisen asettelutekniikan ansiosta. Bootstrap käyttää ruudukkoasettelun toteuttamiseen CSS Flexbox ominaisuuksia (Bootstrap 2019).

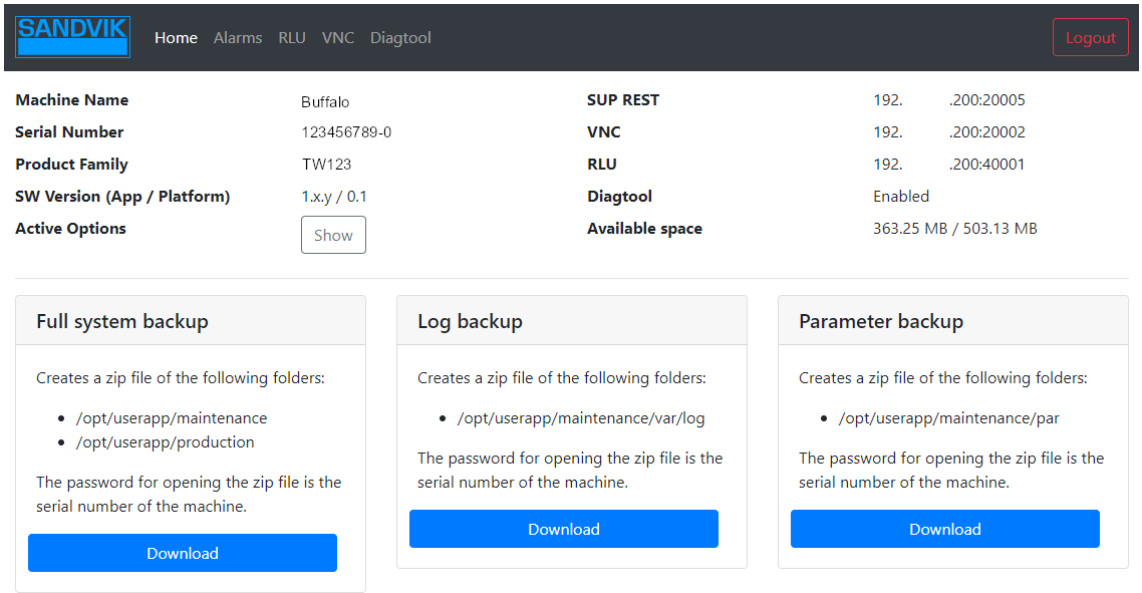
## 4.2 Ominaisuudet

Käyttöliittymän kolme pääominaisuutta ovat varmuuskopioiden lataaminen, hälytyslokien tarkastelu ja kaivoskoneen näytön tarkastelu ja ohjaaminen. Käyttöliittymässä on myös mahdollista tarkastella erään järjestelmään kytketyn moduulin käyttöliittymää, jos sellainen on kytketty. Näiden lisäksi kaivoskoneen erilaisia parametreja on mahdollista tutkia erillisen diagnostiikkatyökalun avulla.

### 4.2.1 Varmuuskopiot

Sovelluksen pääasiallinen käyttötarkoitus on ottaa varmuuskopioita talteen kaivoskoneesta. Tästä syystä myöskin varmuuskopioiden ottaminen laitettiin sovelluksen etusivulle (kuva 10) muiden perustietojen yhteyteen. Varmuuskopioita voi ottaa kolmea erilaista:

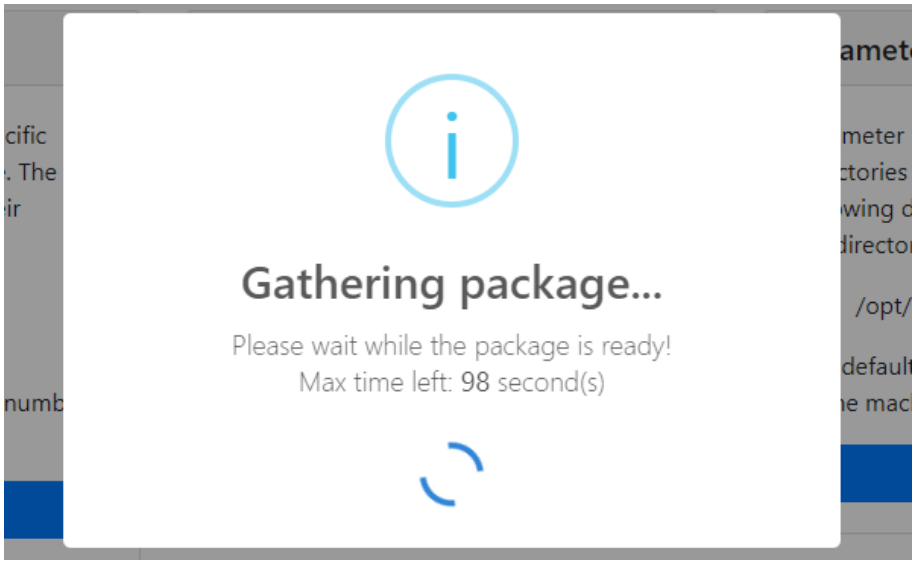
- Täydellinen, joka sisältää kaiken tarvittavan tiedon.
- Lokitiedostot, jotka sisältävät varmuuskopiot vain erilaisista lokeista.
- Parametrit, jotka sisältävät erinäisiä asetustiedostoja.




KUVA 10. Käyttöliittymän kotinäkymä

Varmuuskopioiden lataaminen aukaisee ponnahdusikkunan (kuva 11), jonka aikana käyttäjä ei voi tehdä mitään. Huonojen yhteyksien ja tiedostojen

mahdollisen keräämisen vuoksi lataus saattaa kestää jopa useita minuutteja. Tänä aikana estetään käyttäjän poistuminen sivustolta ennen varmuuskopion suorittamista ja lataamista. Yleisimmissä käyttötapauksissa etähallintasovellus sijaitsee eri laitteella kuin varsinainen varmuuskopioitava laite. Tässä tapauksessa tiedostot joudutaan ensin siirtämään kohdelaitteelta etähallintasovelluksen laitteelle ja tämän jälkeen vasta pakkaamaan ne. Tiedostojen siirto tapahtuu lähiverkossa Ethernet-tekniikalla, mutta tiedostoja voi olla paljon, jolloin siirtokin voi kestää useita kymmeniä sekunteja.



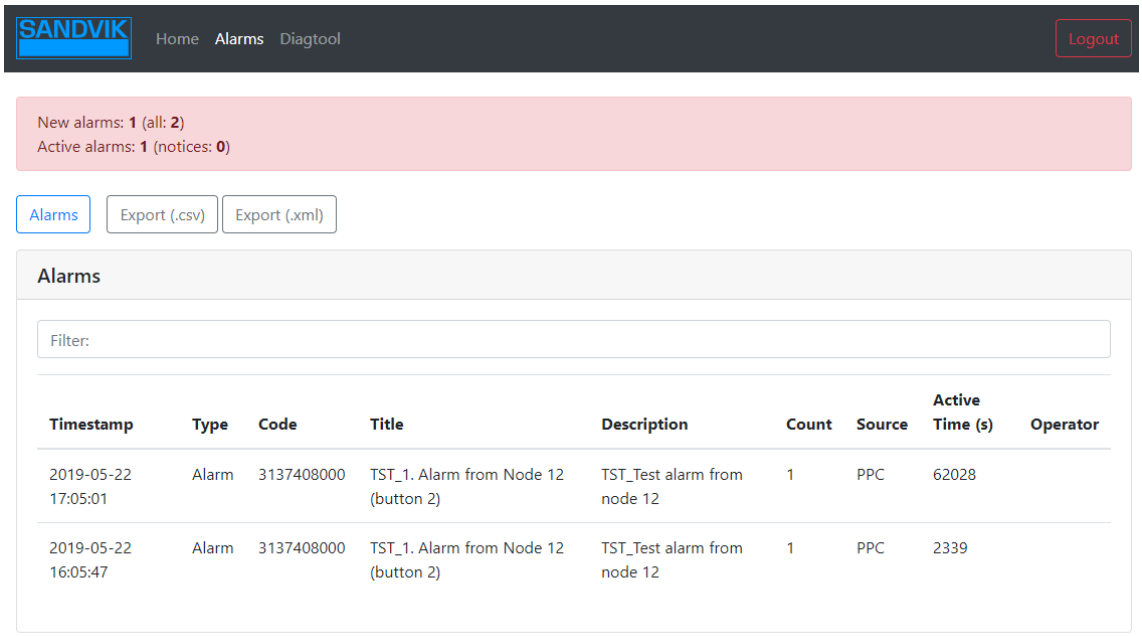

KUVA 11. Varmuuskopioinnin ponnahdusikkuna

Varmuuskopiot tallennetaan laitteella väliaikaiseen kansioon, jonka sisältö tyhjennetään aina jokaisen käynnistyksen yhteydessä. Näin vältytään loputtomasti kasvavilta ja tilaa vieviltä ylimääräisiltä varmuuskopioilta. Tästä huolimatta runsas varmuuskopioiden lataaminen saattaa täyttää laitteen tallennustilan, jolloin käyttäjää ohjeistetaan käynnistämään kaivoskone uudelleen tai poistamaan varmuuskopioita käsin, jos hänellä on siihen oikeudet ja mahdollisuus.

### 4.2.2 Hälytysloki

Käyttöliittymän toinen tärkeä tehtävä on kaivoskoneen mahdollisten hälytysten tarkastelu. Kaivoskone lähettää paljon hälytyksiä erilaisista asioista kuten

polttoaineen loppumisesta, moottorin häiriöstä tai jarrujen epäkunnosta. Hälytyslokin (kuva 12) avulla käyttäjä voi tarkastella hälytysten aikaleimoja, tyyppiä, koodia ja kuvausta. Käyttäjän on myös mahdollista ladata täydellinen hälytysloki joko .csv- tai .xml-tiedostomuodossa.




KUVA 12. Käyttöliittymän hälytysten tarkastelunäkymä

### 4.2.3 VNC-protokolla

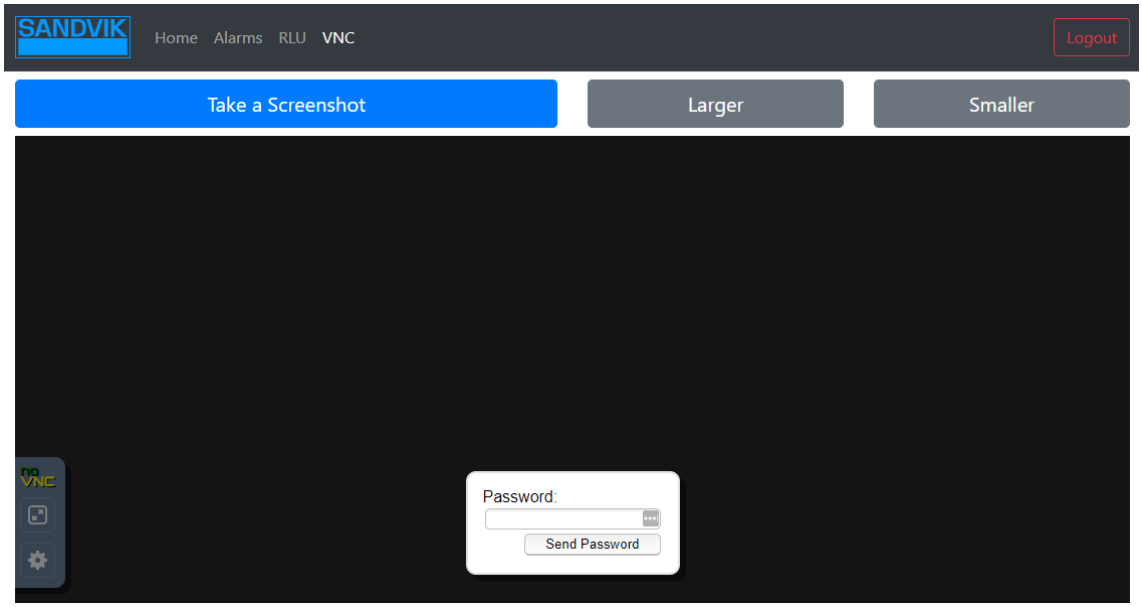
VNC-protokolla mahdollistaa laitteiden graafisen etäkäytön. Protokollan avulla toista tietokonetta voidaan ohjata etänä internet-yhteyden avulla. Etälaitteen käyttöliittymä avautuu yleensä lähes identtisenä ohjaavan käyttäjän näyttöpäätteelle. VNC:tä voidaan käyttää toisen tietokoneen ohjaamisen lisäksi myös hallittavan etälaitteen ruudun tarkasteluun. Tällöin käyttäjä pystyy ainoastaan seuraamaan ruudulla tapahtuvia asioita eikä pysty itse liikuttamaan esimerkiksi kursoria tai kirjoittamaan mitään.

Kaivoskoneista löytyy VNC-palvelin, joten niiden etähallinta on mahdollista toteuttaa. Koska kyseessä on web-sovellus, on valmiita VNC-kirjastoja rajallinen määrä. Pienen tutkiskelun jälkeen ainoaksi varteenotettavaksi vaihtoehdoksi jäi avoimeen lähdekoodiin perustuva noVNC. noVNC tarjoaa jo itsessään HTML-mallisivuston ja JavaScript-kirjaston, jolla VNC-protokollaa voidaan käyttää

selainsovelluksesta. Vaikka kyseessä on avoimen lähdekoodin kirjasto, sitä ei lähdetty tällä kertaa muokkaamaan. Kirjaston mukana tullut HTML-mallisivusto integroitiin osaksi etähallintatyökalua HTML:n iframe-elementillä.

VNC-protokollan data liikkuu tässä tapauksessa WebSocket-protokollan avulla. Siinä käyttäjän selain avaa jatkuvan tunneliyhteyden hallittavaan laitteeseen ja data liikkuu epäsäännöllisin aikavälein edestakaisin. Datan määrän ollessa suurta kaivoskoneiden huono internet-yhteys tulee rajoittavaksi tekijäksi VNC:n käytettävyyden kanssa. Vaikka protokolla lähettääkin vain kohdelaitteella muuttuneet näytön osaset pieninä kuvina, voi datan määrä olla liian suurta hitaalle GSM-yhteydelle. Hidas yhteys aiheuttaa huonosti ja isolla viiveellä päivittyvää kuvaa, jota yritetään automaattisesti kompensoida alentamalla siirrettävän kuvan tarkkuutta ja värisyvyyttä.

Kaivoskoneiden VNC-palvelimet on suojattu salasanalla, jota ilman VNC-yhteyden ottaminen koneeseen ei ole mahdollista. Etähallintasovelluksen VNC-käyttöliittymä (kuva 13) kysyy käyttäjältä erillistä salasanaa kyseisen kaivoskoneen yhteyden avaamista varten. Näin estetään myös tahattomien henkilöiden pääsy tarkastelemaan ja ohjaamaan kaivoskoneen näyttöpäätettä. Tämä siksi, että itse näyttöpäätteestä on jo mahdollista tehdä peruuttamattomia asioita, joiden korjaaminen saattaisi aiheuttaa isoja taloudellisia vahinkoja.




KUVA 13. Käyttöliittymän VNC-näkymä

Kuten käyttöliittymäkuvasta nähdään, on kohdelaitteen ruudulta mahdollista ottaa näyttökaappauksia ja tallentaa ne käyttäjän laitteen muistiin. Kuvakappaukset mahdollistavat ruudulla tapahtuvien asioiden tallentamisen myöhempää tarkastelua varten. Käyttöliittymästä löytyy myös erilliset painikkeet ruudun koon muuttamiseen kohdelaitteeseen sopivaksi. Osa kohdelaitteiden näyttöpäätteistä on kooltaan ja resoluutioltaan pieniä. Joissakin näytöissä näyttöpäätteen tuumakoko nousee huomattavasti ja samassa suhteessa kasvaa myös pikselien määrä eli resoluutio.

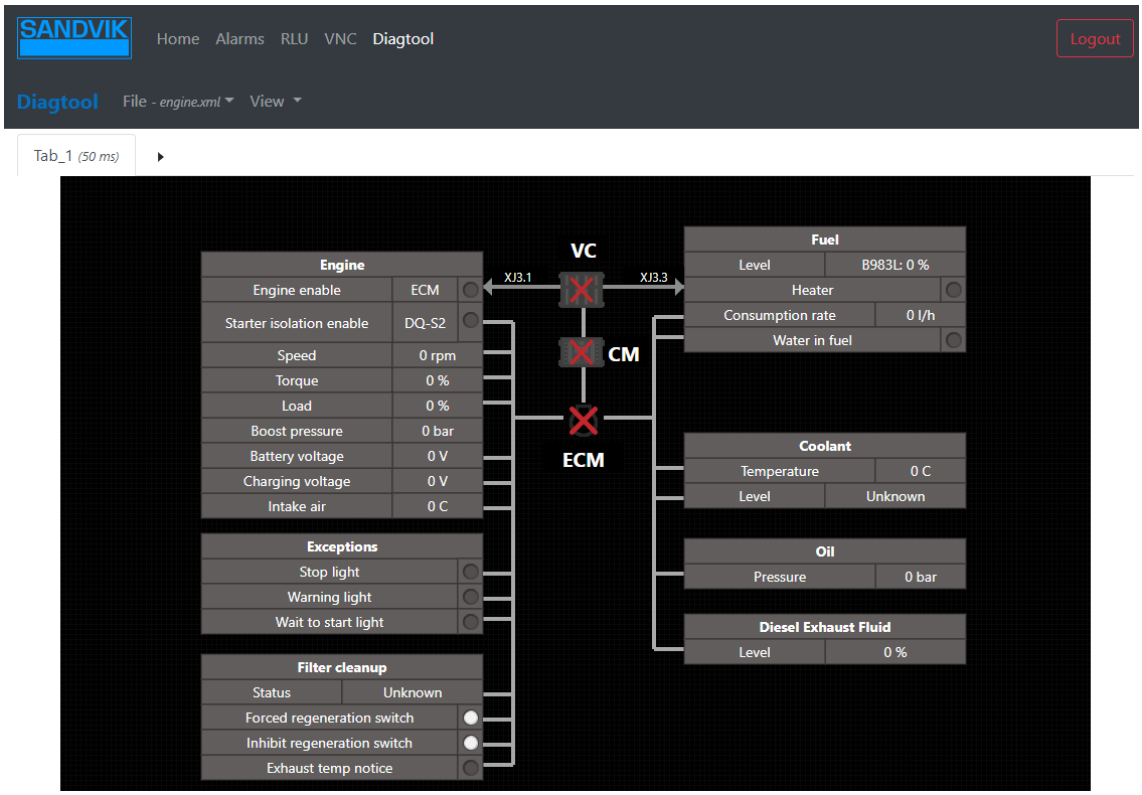
### 4.2.4 Diagnostiikkatyökalu

Kaivoskoneet ovat täynnä erilaisia parametreja, antureita ja sensoreita. Etähallintaohjelmasta voidaan tarkastella näiden arvoja ja tilaa erillisen työkalun avulla. Diagnostiikkatyökalu on toteutettu kokonaisuudessaan toisessa projektissa, mutta sen integroiminen osaksi etähallintatyökalua on mutkatonta. Tämä johtuu siitä, että diagnostiikkatyökalu on vain yksi Angularilla tehty moduuli, jonka lisääminen muihin sovelluksiin on helppoa. Diagnostiikkatyökalun näkymiä voi ladata käyttäjän omalta koneelta tai vaihtoehtoisesti valita kaivoskoneessa jo valmiina olevia näkymiä.

Näkymällä on helppo tarkistaa erilaisten muuttujien, esimerkiksi polttoainemäärän, arvoja. Näkymät voivat olla monimutkaisia ja kattaa esimerkiksi kokonaisen osion kaivoskoneesta. Moottorin diagnostiikkanäkymällä voidaan tarkastella moottorin kierroslukua, vääntöä ja latausjännitettä. Kaivoskoneita huoltaville asiantuntijoille nämä näkymät ovat helppo ja nopea tapa selvittää koneen mahdollisia ongelmia. Etähallintatyökalussa tämä ominaisuus mahdollistaa joidenkin ongelmien selvittämisen etänä sen sijaan että asiantuntijan pitäisi matkustaa kohteeseen jonnekin puolelle maailmaa todetakseen saman ongelman.

Diagnostiikkatyökalu toimii myös WebSocket-protokollan avulla, mutta datan määrä on huomattavasti pienempää kuin VNC-protokollan kanssa. Koneen eri parametrien arvoja lähetetään tasaisin väliajoin ja tällöinkin lähetetään vain muuttuneet arvot. Tällä säästetään tietoliikenteen määrää ja mahdollistetaan työkalun toimiminen huonoillakin yhteyksillä. Esimerkkikuvassa (kuva 14)

parametrien päivitystaajuus on 20 hertsiä eli arvot muuttuvat aina 50 ms välein, yhteensä 20 kertaa sekunnissa. Yhden arvopäivityksen datan suuruus vaihtelee 50 – 800 tavun välillä.




KUVA 14. Käyttöliittymän diagnostiikkatyökalu

## 4.3 Tietoturva

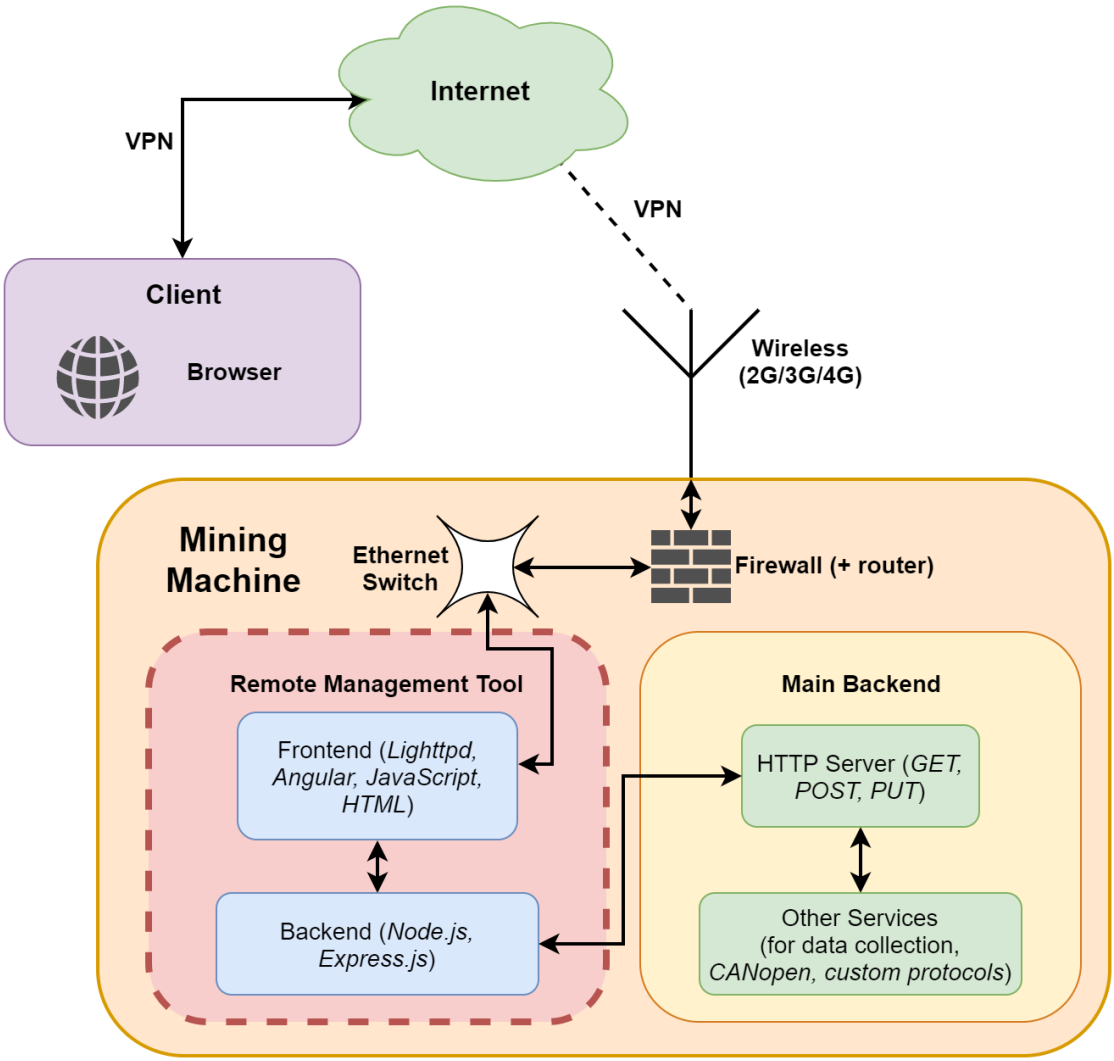
Käyttöliittymän tietoturvaan on panostettu paljon. Käyttäjien pääsy liittymän eri osioihin ilman onnistunutta kirjautumista on estetty kokonaan. Kirjautuminen tapahtuu palvelinpuolta vasten. Kirjautumisen yhteydessä vastaanotetaan avain evästeenä, joka tallennetaan käyttäjän selaimen muistiin, josta se lähetetään aina jokaisen tulevan pyynnön yhteydessä. Evästeet (englanniksi cookies) ovat pieniä paloja dataa, jotka tallennetaan käyttäjän koneelle selaimen toimesta. Evästeet helpottavat käyttäjää sovellusten ja sivustojen käyttämisessä. Ilman evästeitä käyttäjä joutuisi jokaisen toiminnon yhteydessä varmentamaan itsensä uudelleen, mikä olisi käytettävyyden kannalta todella huono käytäntö. Jos käyttäjä muuttaa käsin selaimen muistista löytyvää evästettä, sovellus kirjaa käyttäjän automaattisesti ulos ja estää pääsyn resursseihin ennen uutta

kirjautumista. Jokainen tässä sovelluksessa luotu ja käytettävä eväste on henkilökohtainen ja ainutlaatuinen, joten evästeen muuttaminen edes yhden merkin osalta tekee siitä viallisen eikä sitä voida enää käyttää varmentamiseen. Sovelluksen evästeisiin tallentama varmennusavain on JSON Web Tokenilla luotu ja allekirjoitettu.

Kaivoskoneet ovat yhteydessä internetiin, mutta kaikki liikenne niihin on estetty ilman toimivaa VPN-yhteyttä. Pelkästään jo tämä käytäntö rajoittaa pääsyn koneisiin vain luvallisille henkilöille. Tästä huolimatta käyttöliittymällä estetään vielä mahdolliset kirjautumisyritykset tapauksessa, jossa luvaton henkilö on onnistunut pääsemään suoraan koneen lähiverkkoon kiinni. Kyseinen tapaus on erittäin epätodennäköinen, mutta silti otettava huomioon.

# 5 PALVELINPUOLI

Palvelinpuolen (englanniksi backend) tehtävänä on toimia linkkinä käyttöliittymän ja kaivoskoneen välillä (kuvio 7). Palvelin tarjoaa käyttöliittymälle tietoa kaivoskoneen parametreista ja tämän lisäksi varmentaa käyttäjän. Etähallintatyökalun palvelinpuoli on toteutettu Express.js-sovelluskehyksellä, joka pyörii Node.js-ajoympäristössä. Kyseiseen toteutukseen päädyttiin sen jälkeen, kun alkuperäinen Django-sovelluskehys ei toiminut kaikissa ympäristöissä. Django olisi tarjonnut samat mahdollisuudet ja toiminnallisuudet kuin nykyinen toteutus. Ainoana erona olisi ollut ohjelmointikieli, joka Djangon tapauksessa olisi ollut Python.




KUVIO 7. Arkkitehtuurikuva tietojen siirtymisestä kaivoskoneesta käyttöliittymään

Tässä sovelluksessa palvelinpuoli toimii suurimmalta osin vain välittäjänä kaivoskoneen omalle palvelinpuolen toteutukselle. Käyttäjältä on myös estetty pääsy suoraan kaivoskoneen palvelinpuoleen. Etähallintasovelluksen käyttöliittymä hakee tietonsa palvelinpuolelta, joka puolestaan hakee tietonsa suoraan kaivoskoneen palvelimelta ja välittää ne eteenpäin käyttöliittymälle käyttäjän saataville. Etähallintasovelluksen palvelinpuoli avaa siis kaivoskoneen palvelinpuolelta vain ne tiedot ja päätepisteet, joita käyttöliittymä tarvitsee. Palvelinpuoli myös suodattaa käytössä olevien päätepisteiden tiedoista vain oleelliset tiedot siirrettävän datan vähentämiseksi.

Kaivoskoneen palvelinpuoli taas kerää tietonsa suoraan koneen lähiverkosta ja CANopen-väylästä. CANopen-protokolla soveltuu erinomaisesti kulkuvälineiden ja koneiden eri moduulien tiedonsiirtoon sen nopeuden ja skaalautuvuuden ansiosta. Väylällä kulkee kaivoskoneen eri antureiden ja moduulien arvoja, joita liikkuu tuhansia. Kaivoskoneen palvelinpuoli tarjoaa lukuisia päätepisteitä eri toimintojen suorittamiseksi. Etähallintasovellus käyttää kuitenkin ainoastaan lukemiseen tarkoitettuja päätepisteitä eikä näin ollen kaivoskoneen tietojen muuttaminen ole mahdollista.

## 5.1 Vaatimukset

Käyttöliittymän tapaisesti myöskään palvelinpuolella ei ole mitään erityisiä vaatimuksia käyttäjien varmentamista lukuun ottamatta. Kaikki muut päätepisteet paitsi kirjautuminen vaativat varmennusavaimen lähettämisen pyynnön yhteydessä. Tällä käyttäjä voidaan aina varmentaa ja varmistua siitä, ettei tietoja pääse ulkopuolisten saataville. Palvelinpuolella ei myöskään tehdä mitään järjestelmäkriittisiä toimenpiteitä, joten niiltäkään osin toteutus ei vaadi erityishuomioita.

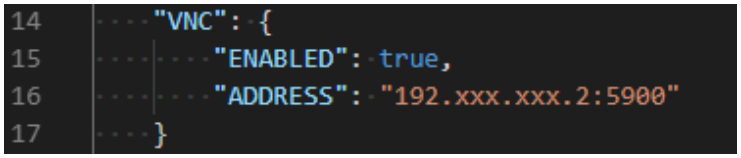
## 5.2 Ominaisuudet

Palvelinpuolella on kolme pääominaisuutta, jotka esitellään tämän luvun kappaleissa. Kirjautuminen, varmuuskopiointi ja tietojen välittäminen eroavat kaikki hieman toisistaan. Jotkin ratkaisut ja tekniikat eivät välttämättä ole kaikkein

tehokkaimpia tai käytännöllisimpiä, mutta tässä tapauksessa ne ovat monesti se ainoa tapa saada kyseinen toiminnallisuus toteutettua. Näiden lisäksi palvelinpuolelta hoidetaan myös etähallintasovelluksen asetusten ja osoitteiden määrittäminen.

### 5.2.1 Asetukset

Etähallintasovelluksen asetukset määritetään palvelinpuolelle, josta ne siirtyvät muiden tietojen mukana käyttöliittymälle. Asetustiedosto noudattaa muodoltaan JSON-tiedostomuotoa (kuva 15). Asetuksista on mahdollista muokata esimerkiksi käyttöliittymän salasanaa, erilaisten moduulien olemassaoloa ja osoitteita sekä määrittää kansioita ja polkuja varmuuskopioille.




KUVA 15. Katkelma asetustiedostosta

Etähallintasovelluksen asennuspaketin mukana tulee asetusten määrittämistä helpottava määritysohjelma, jonka avulla asetusten määrittäminen asennuksen yhteydessä on helppoa. Määritysohjelma on komentoriviltä ajettava työkalu, jonka kysymyksiin vastataan kirjoittamalla ja painamalla lopuksi Enter-näppäintä. Alla oleva esimerkki kuvaa määritysohjelman tulosteita. Esimerkissä vahvennetut tekstit ovat käyttäjän syöttämiä.

```
RemoteManagementTool configuration tool
=======================================
Change predefined values OR leave them
unchanged by pressing 'Enter'

Change REST use HTTPS (current: false): true<Enter>
Changing REST use HTTPS to true
Change VNC enabled (current: true): <Enter>
Using previous: true
```

### 5.2.2 Sisäänkirjautuminen

Käyttäjien varmentamista varten palvelinpuolella on oma päätepiste. Käyttäjä syöttää käyttöliittymän kirjautumissivulla käyttäjänimen ja salasanan, jotka kirjautumisen yhteydessä lähetetään palvelinpuolelle. Jos tunnisteet täsmäävät asetustiedostosta löytyviin tunnisteisiin, käyttäjälle palautetaan vastauksen mukana varmennusavain ja muussa tapauksessa virheviesti. Kirjautumisen jälkeen varmennusavain lähetetään käyttäjän jokaisen tulevan pyynnön mukana ja palvelinpuoli varmentaa tämän avaimen jokaisella kerralla. Avain allekirjoitetaan salaisella merkkijonolla, jolla se on myös myöhemmin mahdollista varmentaa. Sovelluksen salainen merkkijono koostuu useasta erilaisesta ympäristömuuttujasta, joten merkkijonon arvaaminen pääsemättä kohdelaitteelle on käytännössä lähes mahdotonta.

Asetustiedostossa oleva salasana on salattu SHA256-salausalgoritmilla. SHA256-algoritmin tuottama merkkijono on 256 bittiä pitkä. Algoritmi käyttää heksadesimaalista esitystä, jolloin neljä bittiä vastaa yhtä merkkiä ja tästä syntyy 64 merkkiä pitkä merkkijono. Käyttäjältä tuleva salasana salataan samalla algoritmilla ja salattua merkkijonoa verrataan asetustiedostossa olevaan merkkijonoon. Jos merkkijonot täsmäävät, käyttäjä on antanut oikean salasanan. Merkkijonojen erotessa toisistaan voidaan todeta, että käyttäjän syöttämä salasana ei vastaa tallennettua salasanaa.

### 5.2.3 Varmuuskopiot

Palvelinpuolen pääasiallinen tehtävä on kerätä ja tuottaa pakattuja varmuuskopioita kohdelaitteesta. Palvelinpuoli pakkaa tiedostot salasanalla suojattuun pakattuun kansioon, jonka salasana koostuu kohdelaitteen sarjanumerosta. Pakattu kansio sijaitsee väliaikaisessa kansiossa, joka tyhjenee aina laitteen käynnistyksen yhteydessä. Tällä käytännöllä vältytään ylimääräisiltä ja turhaa tilaa vieviltä varmuuskopioilta.

Yleisimmässä käyttötapauksessa kohdelaite sijaitsee eri laitteella, jolloin palvelinpuoli ensin lataa tiedostot omalle laitteelle ja tämän jälkeen vasta pakkaa ne. Näissäkin tapauksissa etähallintatyökalun palvelinpuoli on samassa

lähiverkossa kohdelaitteen kanssa, joten tiedonsiirron ongelmilta vältytään normaalisti. Langallisena samaan lähiverkkoon kytketyt laitteet myös mahdollistavat luotettavat ja kohtalaisen nopeat tiedonsiirtonopeudet. Joissakin varmuuskopiotyypeissä ladattavien tiedostojen koko ei välttämättä ole suuri, mutta tiedostoja saattaa olla useita satoja. Laitteiden pienestä tallennustilasta huolimatta varmuuskopioita voidaan ottaa aina noin 5 – 10 kappaletta ennen kuin tallennustila on täynnä ja laite vaatii uudelleenkäynnistämistä.

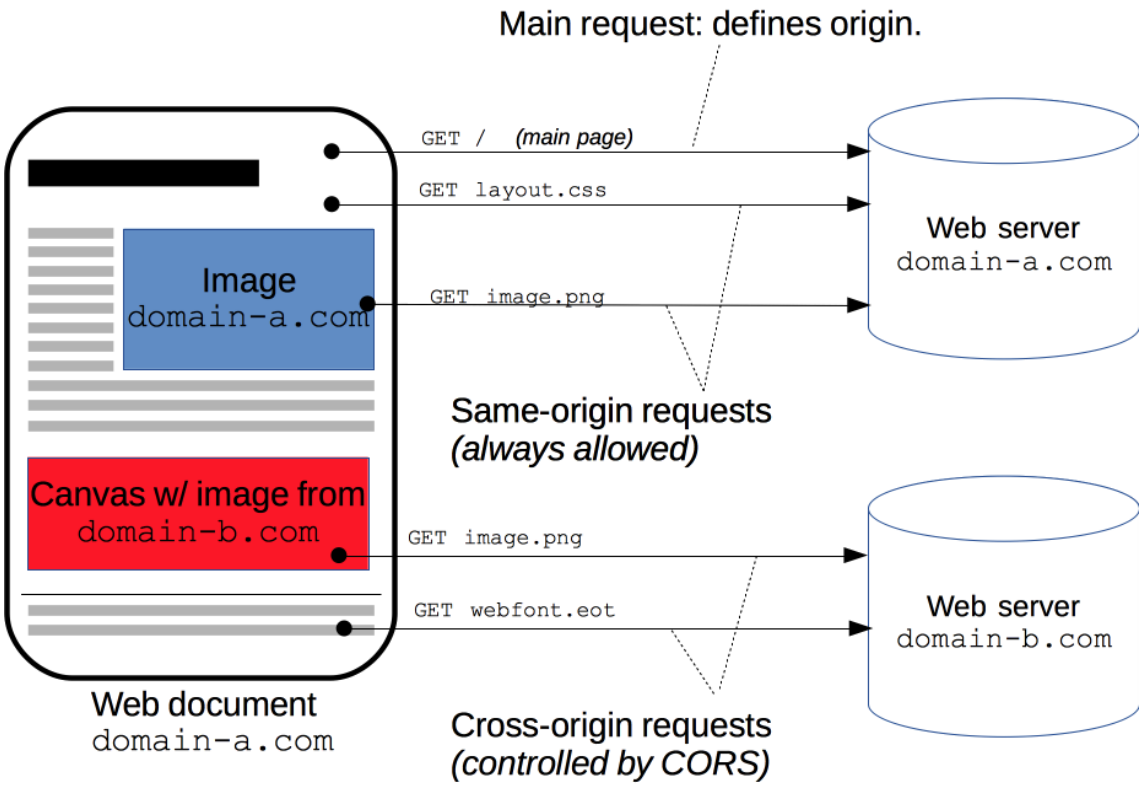
### 5.2.4 Tietojen välitys

Vanhemmissa kohdelaitteissa ei ole mahdollista määrittää CORS-asetuksia (englanniksi Cross-Origin Resource Sharing) kaivoskoneen oman palvelinpuolen päätepisteille. Tällöin käyttäjän selain hylkäisi automaattisesti kaikki pyynnöt ja käyttöliittymä olisi käyttökelvoton. Tämän korjaamiseksi etähallintatyökalun palvelinpuoli toimii tietojen välittäjänä käyttöliittymän ja kaivoskoneen palvelinpuolen välillä (kuva 16). Palvelinpuoli ei välitä CORS-asetuksista ja tällöin se pystyy hakemaan tietoja niiden puuttumisesta huolimatta. Etähallintasovelluksen palvelinpuolelta taas pystytään palauttamaan tarvittavat CORS-asetukset jokaisen vastauksen mukana, jolloin käyttäjän selainkin suostuu vastaanottamaan tietoja.

```
287  /* Return alarms. */
288  router.get('/alarms', function(req, res, next) {
289      axiosInst.get('/api/alarms?format=json')
290          .then(response => {
291              return res.status(200).json({
292                  alarms: response.data
293              });
294          })
295          .catch(error => {
296              console.error(error);
297              return res.status(500).json({});
298          });
299  });
```

KUVA 16. Hälytysten välittäminen

CORS on mekanismi pyyntöjen hallintaan, kun ne tulevat eri lähteestä kuin missä käytettävä sovellus sijaitsee. Samasta alkuperästä tulevat pyynnöt ja niiden vastaukset hyväksytään kaikissa tapauksissa (kuvio 8). Jos sovellus hakee tietoja ulkopuolisesta osoitteesta, tulee vastauksista löytyä ylätunnisteesta tarvittavat tiedot tietojen vastaanottamiselle. Tarvittavien ylätunnistetietojen puuttuessa käyttäjän selain hylkää tietojen vastaanottamisen automaattisesti tietoturvariskin nojalla. Tämä käytäntö suojaa käyttäjiä tapauksissa, joissa web-sovellus sisältää pahalaatuista koodia, jonka tarkoitus on tehdä jotain sopimatonta käyttäjän laitteelle. Tällaista on esimerkiksi haittaohjelmien lataus tai sopimattomien käyttäjätietojen lähettäminen jonnekin toiselle palvelimelle ilman käyttäjän lupaa ja tietoa asiasta.




KUVIO 8. Cross-Origin Resource Sharing, CORS (Mozilla Developer 2019)

### 5.3 Tietoturva

Palvelinpuolen kaikki päätepisteet kirjautumista lukuun ottamatta vaativat käyttäjältä varmennusavaimen pyynnön mukana. Ilman varmennettua avainta käyttäjä ei pysty hakemaan tietoja palvelinpuolelta. Varmennusavain luodaan,

allekirjoitetaan ja varmennetaan JSON Web Tokenin tarjoamilla metodeilla. Avaimen allekirjoitukseen käytetään järjestelmän ympäristömuuttujia, jolloin salaista merkkijonoa ei ole mahdollista löytää esimerkiksi versionhallinnasta. Lisäturvan tuomiseksi käyttäjän varmennusavain on voimassa vain 24 tuntia sen luomisesta.

Palvelinpuolen tietoturvaan on panostettu tarkoituksella, sillä jotkin sieltä saatavat tiedot saattaisivat mahdollisesti sisältää jotain merkityksellistä. Saatavalla tiedolla voisi olla merkittäviä kilpailullisia etuja kolmansille osapuolille. Edistyneiden käytänteiden lisäksi palvelinpuolella on käytetty muita tavallisia ja hyväksi havaittuja keinoja tietoturvan parantamiseksi. Express.js listaa sivuillaan tuotantopalvelimen parhaita tietoturvakäytänteitä, joita etähallintasovelluksen palvelinpuolella noudatetaan sen mukaan, mikä on mahdollista ja tarpeellista.

Käyttäjien ollessa pääasiassa asiakkaan tuotantopuolen omia asiantuntijoita, on palvelinpuolelta silti estetty kaikki yleiset mahdolliset virhetilanteet ja käyttäjien mahdollisesti tekemät virheelliset toiminnot. Virhetilanteissa palvelinpuoli palauttaa aina asianmukaisen virhekoodin ja yleisimmissä tapauksissa myös käyttäjä osaa virhekoodin ja -viestin perusteella tehdä tarvittavia toimenpiteitä tilanteen korjaamiseksi.

# 6 POHDINTA

Kaikki projektin alussa määritellyt ominaisuudet saatiin toteutettua. Lopputuloksena syntynyt etähallintatyökalu täyttää sille ennalta asetetut tavoitteet, joten opinnäytetyön lopputulokseen voi olla tyytyväinen. Angularilla toteutettu käyttöliittymä ja Express.js-kehyksellä rakennettu palvelinpuoli sopivat tekniikoiltaan erinomaisesti vastaamaan työkalun käyttötarkoitusta. Tekniikat ja kehykset olivat suurimmalta osalta jo ennalta tuttuja, joten käyttöliittymän ja palvelinpuolen toteutus onnistuivat mutkitta kehittäjien omien teknisten dokumentaatioiden avulla.

Mielenkiintoisimmaksi ja kohtalaisesti aikaa vieneeksi ominaisuudeksi muodostui kirjautuminen ja käyttäjien määrittäminen käyttöliittymälle. Varsinkin tietoturvan säilyttämistä kaikissa vaiheissa joutui välillä miettimään useampaankin otteeseen. Varmennusavainten varmistaminen ja niiden väärinkäyttämisen estäminen aiheuttivat omia hankaluuksiaan. Tietoturvan ohessa piti käyttökokemus säilyttää inhimillisellä tasolla. Yleensä tietoturvan lisääminen vain heikentää käyttökokemusta ja toisinpäin. Kaivoskoneiden käyttäjät ovat pääasiassa asiantuntijoita ja koneet VPN-yhteyksien takana, mutta siitä huolimatta varsinkin jatkokehityksessä tulevien käyttäjätasojen ylläpitäminen oli ensisijaisen tärkeää. Käyttäjän ei pidä pystyä millään tavalla onnistua saamaan itselleen korkeampaa käyttäjätasoa kuin hänelle on asetettu. Tässä tavoitteessa onnistuttiin myös hyvin. Sovellus kirjaa käyttäjän ulos heti, jos varmennusavainta on yritetty muuttaa.

Internet-yhteysongelmat mietityttivät projektin alkaessa. Kenttätestien ja alustavienkin testien perusteella yhteyksien hitaus vaikutti ominaisuuksien toimintaan odotetulla tavalla. Pääasialliset ominaisuudet kuten varmuuskopioiden ottaminen ja hälytyslokin tarkastelu kuitenkin vaativat osaltaan vähän dataa tai datan siirtonopeudella ei ole merkitystä toimivuuden kanssa. Yhteyksien kehittyessä jatkuvasti nämä ongelmat kuitenkin harvinaistuvat koko ajan. Kaivoskoneiden oman tiedonsiirron tarvekin kasvaa jatkuvasti ja tämä erityisesti pakottaa kaivoksia panostamaan myös langattomien tiedonsiirtoteknologioiden parantamiseen kaivosympäristöissään.

Etähallintatyökalun jatkokehitys jatkuu pienen testivaiheen jälkeen käyttäjiltä kerätyn palautteen perusteelta. Erityisesti käyttöliittymän tukea useammalle kielelle toivottiin jo alkuvaiheessa. Ominaisuuden toteutus jätettiin kuitenkin toteuttamatta ensimmäisessä versiossa ratkaisemattomien taustatekijöiden vuoksi. Lisäksi laitetuen lisääminen on ensimmäisiä jatkokehityksen kohteita. Myöskin kaivoskoneiden palvelinpuoli on laajentunut projektin aikana ja se mahdollistaisi jälleen uusien ominaisuuksien lisäämisen osaksi etähallintatyökalua.

Kokemuksena projekti oli erittäin opettavainen. Työkalua kehittäessä tuli tutustuttua uusiin sovelluskehyksiin, joista voi olla apua tulevaisuuden projekteissa. Projektia ja sen lopputuloksena syntynyttä sovellusta voidaan käyttää esimerkkinä ja apuna seuraavia projekteja ja vastaavia sovelluksia suunniteltaessa ja kehittäessä.

**LÄHTEET**

Angular. 2019. Introduction. Luettu 11.9.2019. https://angular.io/docs

Atlassian. 2019. What is Continuous Integration. Luettu 12.9.2019. https://www.atlassian.com/continuous-delivery/continuous-integration

Bitwise Oy. Referenssit – Sandvik Mining and Rock Technologies. Luettu 25.8.2019. https://bitwise.fi/referenssit/sandvik-mining-and-rock-technologies/

Bootstrap. 2019. Grid system. Luettu 12.9.2019. https://getbootstrap.com/docs/4.3/layout/grid/

CrossControl. 2018. About Group. Luettu 1.9.2019. https://crosscontrol.com/en-us/about/group

Epec Oy. 2019. Company. Luettu 1.9.2019. https://epec.fi/company/

Express.js. 2019. Express. Luettu 3.8.2019. https://expressjs.com/

ExpressVPN. 2019. What is a VPN? Luettu 1.9.2019. https://www.expressvpn.com/what-is-vpn

Gerrit Code Review. 2019. Gerrit's history. Luettu 11.9.2019. https://www.gerritcodereview.com/about.html

Git. 2019. Alkusanat - Versionhallinnasta. Luettu 1.9.2019. https://git-scm.com/book/fi/v1/Alkusanat-Versionhallinnasta

GitHub. 2019. noVNC. Luettu 13.9.2019. https://github.com/novnc/noVNC

Internet Engineering Task Force. 2015. RFC 7519. Luettu 11.9.2019. https://tools.ietf.org/html/rfc7519

Jenkins. 2019. What is Jenkins? Luettu 12.9.2019. https://jenkins.io/doc/

JSON Web Tokens. 2019. JWT. Luettu 11.9.2019. https://jwt.io/

Lighttpd.net. 2019. Lighttpd. Luettu 28.10.2019. https://www.lighttpd.net/

Linux.fi. 2015. VNC. Luettu 25.8.2019. https://www.linux.fi/wiki/VNC

Mozilla Developer. 2019. Cross-Origin Resource Sharing (CORS). Luettu 14.9.2019. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Mozilla Developer. 2019. JavaScript. Luettu 1.9.2019. https://developer.mozilla.org/en-US/docs/Web/JavaScript

Node.js. 2019. About Node.js. Luettu 1.9.2019. https://nodejs.org/en/about/

npm. 2019. About npm. Luettu 11.9.2019. https://docs.npmjs.com/about-npm/

Python. 2019. About. Luettu 1.9.2019. https://www.python.org/about/

TypeScript. 2019. TypeScript. Luettu 1.9.2019. https://www.typescriptlang.org/

Sandvik Group. 2019. Business areas. Luettu 25.8.2019. https://www.home.sandvik/en/about-us/our-company/business-areas/

Sandvik Mining and Rock Technology. 2019. About us. Luettu 25.8.2019. https://www.rocktechnology.sandvik/en/about-us/

Stack Overflow. 2019. Developer Survey Results. Luettu 11.9.2019. https://insights.stackoverflow.com/survey/2019

Visual Studio Code. 2019. Getting Started. Luettu 11.9.2019. https://code.visualstudio.com/docs

V8. 2019. What is V8? Luettu 1.9.2019. https://v8.dev/

Yle. 2019. Sandvik aloittaa yt-neuvottelut. Luettu 25.8.2019. https://yle.fi/uutiset/3-10932813

## LIITTEET

Liite 1. Skripti etähallintatyökalun hallitsemiseen

```
# Init.d script to handle Sandvik Remote Connection Tool

APPNAME=RemoteConnectionTool
APP_DIR="/opt/user/remote-connection-tool/"
FRONTEND_EXEC="$APP_DIR/frontend_runtime.sh"
FRONTEND_PID_FILE="$APP_DIR/frontend.pid"
BACKEND_EXEC="$APP_DIR/backend_runtime.sh"
BACKEND_PID_FILE="$APP_DIR/backend.pid"

if [ -e "/etc/default/$APPNAME" ]
then
    . /etc/default/$APPNAME
fi

start() {
    # Start backend (Nodejs)
    $BACKEND_EXEC
    # Start frontend (Lighttpd)
    $FRONTEND_EXEC
}

stop() {
    # Stop Lighttpd
    if [ -f "$FRONTEND_PID_FILE" ]; then
        kill $(cat "$FRONTEND_PID_FILE")
        rm -f $FRONTEND_PID_FILE
    else
        echo "No PID file found (frontend)"
    fi
    # Stop Nodejs
    if [ -f "$BACKEND_PID_FILE" ]; then
        kill $(cat "$BACKEND_PID_FILE")
        rm -f $BACKEND_PID_FILE
    else
        echo "No PID file found (backend)"
    fi
}

restart() {
    stop
    start
}
```

(jatkuu)

```
status() {
    [ -f "$BACKEND_PID_FILE" ] && echo "RCT backend is running" || echo
"RCT backend is NOT running (no PID file found)"
    [ -f "$FRONTEND_PID_FILE" ] && echo "RCT frontend is running" || echo
"RCT frontend is NOT running (no PID file found)"
}

case "$1" in
start)
    start
    ;;
stop)
    stop
    ;;
force-reload|restart)
    restart
    ;;
status)
    status
    exit 0
    ;;
*)
    echo "Usage: /etc/init.d/$APPNAME {start|stop|restart|force-
reload|status}"
    exit 1
    ;;
esac

exit 0
```