Ville Harjunen

Service Suite - Huoltosovelluksen toteutus hybridisovelluksena

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

11.11.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Ville Harjunen Service Suite - Huoltosovelluksen toteutus hybridisovelluksena 30 sivua + 1 liite 11.11.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Simo Silander

Tässä insinöörityössä kuvataan huoltosovelluksen toteutusta lääketieteen alan yritykselle. Huoltosovellus toteutettiin asiakkaan tarpeeseen sähköistää MEG-laitteen huoltoprosessi.

Työn tavoitteena oli toteuttaa sellainen ratkaisu asiakkaan huoltoprosessin raportointiin, jossa paperille täytettävät huoltoraportit korvattaisiin sähköisellä järjestelmällä, ja huoltoraporttien tallennus tapahtuisi pilvipalveluun. Tavoitteena oli myös helpottaa huoltomiesten työtä yksinkertaistamalla huoltoon kuuluvia tehtäviä.

Työssä toteutettiin moderneihin web-sovelluksiin perustuva sovelluspaketti käyttäen vapaan lähdekoodin JavaScript-kirjastoja, kuten Node.js:ää ja AngularJS:ää. Työssä tutustuttiin myös hybridisovelluksiin ja niiden toteutukseen sekä Amazonin tarjoamiin pilvipalveluihin Amazon Elastic Compute Cloud -pilvipalvelimiin ja Amazon Elastic Block Store -tallennuspalveluun.

Huoltosovellus toteutettiin hybridisovelluksena Electron-ohjelmistokehyksellä ja sen web-sovellus modernina web-sovelluksena. Huoltosovelluksen tueksi toteutettiin palvelinohjelma ja pystytettiin sopivanlainen pilvipalvelin. Huoltosovelluksen tuottamien tulosten katselemiseen ja huoltosovelluksen päivityksen hallintaan toteutettiin erillinen hallintasovellus. Hallintasovelluksen web-sovellus toteutettiin samoin teknologioin kuin huoltosovelluksen web-sovellus.

Työn lopussa kerrotaan projektin onnistumisesta ja hybridisovellusten sopivuudesta tällaiseen projektiin.

Avainsanat	Electron, hybridisovellus, AngularJS, web-sovellus

Author Title Number of Pages Date	Ville Harjunen Service Suite – Development of Maintenance Software as Hybrid application 30 pages + 1 appendix 11 November 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Simo Silander, Senior Lecturer

This study describes the development of a maintenance software for a medical company. The maintenance software was developed for the customer company's need to digitalize the maintenance workflow of MEG devices.

The aim of the study was to develop a solution to digitalize the maintenance workflow in a such a way that hand filled maintenance reports would be replaced with a computer program and the reports would be saved in the cloud. Also, the maintenance workflow of a service person would be improved by making some of the steps in the maintenance workflow easier.

A software package based on modern web applications was implemented. Web applications were developed with open source JavaScript libraries such as Node.js and AngularJS. The study explores hybrid web applications and Amazon Elastic Compute Cloud and Amazon Block Store, which are components of Amazon cloud platform.

The maintenance software was developed as a hybrid web application based on the Electron framework. The web application of the maintenance software was developed as a modern web application. The server application was developed and installed to a suitable cloud server. For the need to review the reports generated by the maintenance software and management of updates, the admin software was developed. The web application of the admin software was based on the same technologies as the web application of the maintenance software.

The study presents the result of the project and gives some thoughts on the suitability of hybrid application technologies in this kind of a project.

Keywords	Electron, hybrid application, AngularJS, web application

Sisällys

Lyhenteet ja termit

JSON	JavaScript Object Notation, avoimen standardin tiedostomuoto.
MEAN	MongoDB, AngularJS, Express.js, Node.js.
AngularJS	Googlen valmistama vapaan lähdekoodin JavaScript-ohjelmistokehys.
Node.js	Vapaan lähdekoodin JavaScript-ohjelmistokehys palvelinohjelmointiin.
MongoDB	Vapaan lähdekoodin JSON-pohjainen NoSQL-tietokanta.
AWS	Amazon Web Services, Amazonin tarjoama pilvipalvelinalusta.
MEG	Magnetoenkefalografia, aivomagneettikäyrä.
DACQ	Data acquisition computer, MEG-laitteeseen kuuluva tietokone.

1 Johdanto

Insinöörityö tehdään Elekta Oy:n tilauksesta. Työssä toteutetaan sovelluspaketti kohdeyrityksen huolto- ja testaustoiminnan sähköistämiseksi. Sovelluspaketti on neljästä osasta koostuva työpöytäsovellus ja pilvipalvelu, Service Suite. Sovelluksesta on tehty aiemmin prototyyppi, jonka perustuksista tätä sovellusta ryhdyttiin suunnittelemaan ja toteuttamaan.

Sovelluspaketin tarkoituksena on nykyaikaistaa kohdeyrityksen huoltotoimenpiteitä. Nykyaikaistaminen tapahtuu sähköistämällä työvaiheiden tulosten kirjaaminen ja helpottamalla MEG-laitteelle tehtäviä mittauksia. Huoltotoimenpiteiden tulokset myös kerätään keskitetysti yhteen pilvipalveluun, joka mahdollistaa tulosten tarkastelemisen jälkeenpäin ja vertailun.

Service Suiten tekninen toteutus tehdään käyttäen moderneja web-teknologioita. Sovelluspaketin normaalikäyttäjälle eli huoltomiehelle näkyvin osa, Service Suite Client, toteutetaan hybridityöpöytäsovelluksena Electron-ohjelmistokehyksen päälle modernina web-sovelluksena.

Tämän insinöörityön tavoitteena on kertoa Service Suiten suunnittelusta ja kehityksestä sekä sen kehitykseen käytetyistä teknologioista. Lisäksi työssä pohditaan hybridityöpöytäsovellusten nykytilannetta ja tulevaisuuden mahdollisuuksia.

2 Pilvipalvelut

Perinteisesti verkkopalveluita rakentaessa vaihtoehdot palvelinarkkitehtuuriksi ovat olleet omat palvelimet, webhotellit tai virtuaalipalvelimet. Webhotellit ovat palveluita, jossa asiakas vuokraa kiintolevytilaa palveluntarjoajalta. Webhotellit eivät itsessään sisällä laskentatehoa, joten monet webhotellien palveluntarjoajat tarjoavatkin niiden rinnalla virtuaalipalvelimia. Virtuaalipalvelimet ja omat palvelimet eivät käytön suhteen eroa toisistaan juuri ollenkaan vaan niiden suurimpana erona on rautatason ylläpitotoimet. Omia palvelimia joutuu ylläpitämään itse, kun taas virtuaalipalvelimien ylläpito tapahtuu palveluntarjoajan toimesta.

Pilvipalvelu voi yksinkertaisimmillaan olla todella samanlainen kuin virtuaalipalvelin. Pilvipalvelut tarjoavat paljon muita palveluita pelkän palvelimen lisäksi. Pilvipalveluntarjoaja voi esimerkiksi tarjota CDN:n (content delivery network) tai tietokantapalvelimen.

Pilvipalveluntarjoajalta ostettu virtuaalipalvelin on turvallinen ja luotettava vaihtoehto nykypäivänä verkkosovelluksen alustaksi. Pilvipalveluntarjoaja lupaa lähes katkeamattoman saavutettavuuden palvelimille. Pilvipalvelinta pystyttäessä voidaan helposti valita palvelimen fyysinen sijainti, joka edesauttaa käyttäjän ja palvelimen välistä yhteyttä.

Pilvipalveluntarjoajat tarjoavat paljon muutakin kuin pelkkiä palvelimia. Pilvipalvelut tarjoavat esimerkiksi valvontatyökaluja, nimipalvelujärjestelmät ja varmuuskopiointityökalut. Omilla palvelimilla nämä jouduttaisiin itse pystyttämään ja konfiguroimaan.

Näiden yllämainittujen seikkojen myötä tähän projektiin valitaan ulkoisen toimijan tarjoamat pilvipalvelut itse ylläpitämien palveluiden sijaan. Pilvipalveluiden tarjoajia löytyy useita. Ne tarjoavat erisuuruisen määrän ominaisuuksia. Osa tarjoajista tarjoaa todella yksinkertaisesti käyttöönotettavia palveluita, mutta yksinkertaisuuden myötä menetetään hallittavuutta, jota tässä projektissa vaaditaan. Muutaman eri pilvipalvelutarjoajan vertailun voittajaksi selviytyi Amazon ja sen tarjoama Amazon Web Services.

2.1 Amazon Web Services

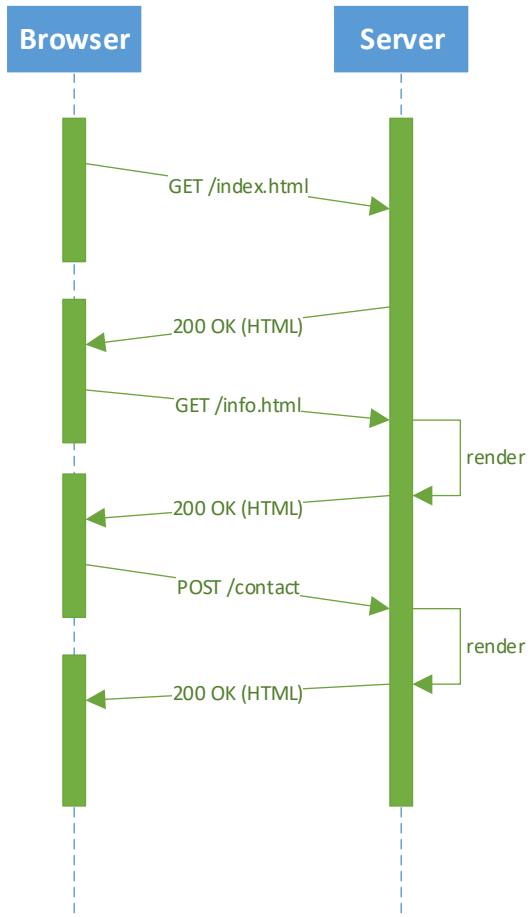
Amazon Web Services, AWS, on selkeä markkinajohtaja pilvipalveluiden tarjoajana. AWS tarjoaa käyttäjälle yhden markkinoiden laajimman paketin erilaisia ominaisuuksia. AWS:n keskeisimmät ominaisuudet kuitenkin ovat sen tarjoamat Amazon EC2 ja Amazon S3.

EC2, Amazon Elastic Compute Cloud on palvelu, joka tarjoaa virtuaalisia palvelininstansseja. Nämä instanssit ovat helposti skaalattavissa tehojensa puolesta, ja niiden pystyttäminen ja ylläpitäminen on helppoa ja nopeaa. Kuitenkin hallinta säilyy. EC2-instanssit ovatkin tämän projektin palvelininfrastruktuurin selkärankana.

Amazon tarjoaa muutamaa erilaista tallennustilaratkaisua hieman erilaisista näkökumista rakennettuna. S3, Amazon Simple Storage Service on objektivarasto (object storage), johon tieto tallennetaan objektimuotoisena. Tämän lisäksi Amazon tarjoaa Amazon Elastic Block Store (EBS) -tallennusratkaisuaan, joka vastaa käytännössä normaalia tallennusmediaa, kuten kiintolevyä. Tähän projektiin valitaan tallennusratkaisuksi EBS, joka sopii paremmin sovelluksen vaatimukseen siitä, että tallennettu data on oltava omassa hallussa ja hallittavissa sekä datan muoto voi vaihdella.

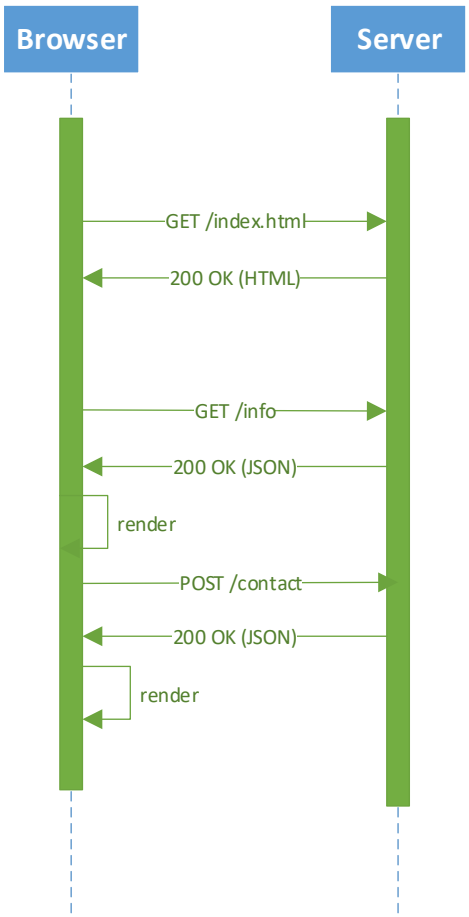
3 Modernit web-sovellukset

Modernit web-sovellukset ovat web-sovelluksia, jotka pohjautuvat uusiin JavaScript-kirjastoihin ja -ohjelmistokehyksiin, kuten AngularJS:ään, Angulariin tai Reactiin. Nämä sovellukset ovat HTML5-sovelluksia tai sivuja, jotka suoritetaan asiakaspäässä, eli käyttäjän selaimessa, palvelimen sijaan.



Kuva 1. Perinteinen verkkopalveluarkkitehtuuri

Modernien web-sovellusten toiminta eroaa huomattavasti perinteisistä web-sovelluksista. Käyttäjän saapuessa perinteiseen web-sovellukseen selain lataa palvelimella renderöidyn näkymän, kuva 1 GET /index.html. Kun käyttäjä klikkaa sivulla jotakin elementtiä, selain pyytää palvelimelta uutta näkymää, jonka palvelin renderöi ja palauttaa (ks. kuva 1 GET /info.html). Tällainen verkkopalveluarkkitehtuuri aiheuttaa paljon ylimääräistä lataamista, kun jokaisesta interaktiosta sivu joudutaan lataamaan uudelleen. Tämä näkyy erityisesti heikkotehoista päätelaitetta ja hidasta internetyhteyttä käyttäville, kun jokainen sivulataus kestää huomattavan kauan. Tämän ongelman vuoksi on kehitetty modernit web-sovellukset.



Kuva 2. Moderni verkkopalveluarkkitehtuuri

Käyttäjän saapuessa moderniin web-sovellukseen selain lataa sovelluksen ohjelmakoodin palvelimelta ja renderöi sivun (ks. kuva 2, GET /index.html). Sovellus jää pyörimään käyttäjän selaimeen ja käyttäjän klikatessa jotakin elementtiä sivustolla, selaimen ei tarvitsekaan pyytää täysin uutta näkymää palvelimelta, vaan se pyytää palvelimelta vain kyseiseen interaktioon liittyvät tiedot ja renderöi käyttäjän päässä uuden näkymän (ks. kuva 2. GET /info). Sovelluksen toimiessa tähän tapaan uusia sivunlatauksia ei tapahdu. Sovelluksen ensilataus voi kestää perinteistä sovellusta hieman kauemmin, mutta interaktiot sovelluksen kanssa tapahtuvat huomattavasti nopeammin.

Moderneja web-sovelluksia kutsutaan usein myös termillä SPA, Single Page Application.

3.1 MEAN

MEAN on kokoelma JavaScript-kirjastoja, jolla voi rakentaa moderneja web-sovelluksia. MEAN koostuu MongoDB-tietokannasta, Express.js-kirjastosta, AngularJS-kirjastosta ja Node.js-sovellusalustasta. Nämä kaikki ovat JavaScriptiä toteuttavia teknologioita ja kirjastoja, joten kaikki ohjelmakoodi sekä palvelin- että selainpäässä on JavaScriptiä. Tämä mahdollistaa yhden ohjelmistokehittäjän kehittää helposti yhtä ohjelmointikieltä osaamalla kokonaisia sovelluksia.

3.2 MongoDB

MongoDB on NoSQL-tietokanta, johon tieto tallennettaan JSON-tyyppisenä. Perinteisempiin SQL-tietokantoihin verrattuna MongoDB eroaa eniten siinä, että siinä ei ole ennalta määrättyä taulukkoskeemaa. MongoDB sopii hyvin skaalautuviin web-sovelluksiin ja erityisesti JavaScript-pohjaisiin sovelluksiin, koska tieto on tallennettu tietokantaan samassa muodossa kuin sitä käsitellään itse sovelluksessa. (1)

Sovellustasolla tietokannassa olevaa tietoa halutaan kuitenkin usein käsitellä ennalta määrätyllä rakenteella, jotta ohjelmointi olisi sujuvampaa ja tiedon muoto pysyisi oikeanlaisena. Tätä tarkoitusta varten on kehitetty erilaisia kirjastoja, joilla taulukkoskeema luodaan sovellustasolla käytettäväksi, mutta tietokanta itse ei sen olemassaolosta tiedä mitään. Node.js-ympäristössä Mongoose on yleisimmin käytössä oleva kirjasto.

3.3 Node.js ja Express.js

Node.js, lyhyemmin Node, on Googlen kehittämään JavaScript-moottoriin V8:n perustuva sovellusalusta. Node mahdollistaa JavaScript-koodin ajamisen palvelimella. Node pitää sisällään suhteellisen alhaisen tason ominaisuuksia, kuten tiedostojen lukemisen. Node.js on nykyään todella suosittu niin pienissä kuin todella suurissakin sovelluksissa sen vaivattoman skaalautumisen vuoksi.

Express.js, lyhyemmin Express, on Node.js-sovelluskehys, jolla toteutetaan sovellusten palvelinpuolen logiikkaa. Expressin perustoiminta on yhdistää polkuja niihin soveltuviin

logiikkafunktioihin. Expressin avulla on yksinkertaista toteuttaa REST-rajapinta tietokannan ja selainsovelluksen välille.

3.4 AngularJS

AngularJS on Googlen kehittämä vapaan lähdekoodin JavaScript-kirjasto, jonka avulla voidaan luoda SPA-sovelluksia. AngularJS on rakennettu JavaScriptillä, joten sitä voidaan ajaa täysin selaimessa. (2)

AngularJS tuo MVC-arkkitehtuurin web-sovelluksiin. Näkymää edustaa DOM, kontrolleria edustaa JavaScript-funktiot ja mallia vastaa data, joka on tallennettuna JavaScript-muuttujiin. MVC-malli tuo kehittämiseen selkeyttä ja helpottaa ohjelmistokehittäjää ymmärtämään sovelluksen rakenteen. Tämä on kehitystä verrattuna vanhempiin ratkaisuihin, joissa MVC-mallin toteutusta ei yleensä ole. (3)

AngularJS toteuttaa komponenttimallia, jossa sovellus paloitellaan useisiin pieniin komponentteihin. Jokainen komponentti toteuttaa oman MVC-mallin eli pitää sisällään kyseiseen komponenttiin liittyvän näkymän, datan ja funktiot. Sovellukset koostuvat yhdistelemällä näitä komponentteja. Komponenttien lisäksi AngularJS tarjoaa mahdollisuuden luoda yhteisessä käytössä olevia funktioita (service, factory, directive, pipe). (2)

Perinteisissä web-sovelluksissa käytetään yksisuuntaista datakytkentää. Yksisuuntaisessa datakytkennässä sivun HTML renderöidään palvelimella, jonka jälkeen se lähetetään selaimelle. AngularJS mahdollistaa kaksisuuntaisen datakytkennän web-sovelluksen käyttöliittymän ja tietorakenteen välille. Tämä tarkoittaa sitä, että näkymää voidaan muokata dynaamisesti ja reaaliaikaisesti. AngularJS tekee tämän automaattisesti ja toteuttaa sen siten, että jos data joko käyttöliittymässä tai tietorakenteessa muuttuu, se päivittää myös toisen vastaamaan muutosta. (3)

Koska AngularJS on Googlen kehittämä, se on saavuttanut suuren yhteisön ympärilleen ja siihen löytyy todella paljon yhteisön kehittämiä kirjastoja ja apuvälineitä. (2)

4 Hybridisovellukset

Hybridisovellus on sellainen sovellus, joka on kehitetty perinteisillä web-teknologioilla joita ovat HTML5, CSS ja JavaScript. Sen jälkeen se on käännetty natiiviksi ohjelmakoodiksi. Hybridisovelluksia on kahdenlaisia mobiili- ja työpöytähybridisovelluksia. Vaikka niillä on paljon yhteistä, eroavat ne silti toisistaan hieman. Hybridisovellukset mobiilissa hyödyntävät Android- ja iOS-käyttöjärjestelmistä löytyvää WebView-teknologiaa. WebView-teknologia on tarkoitettu verkkosisällön näyttämiseen natiivisovelluksen sisällä. Hybridisovellukset työpöydällä eivät voi hyödyntää vastaavanlaista teknologiaa, koska työpöytäkäyttöjärjestelmät eivät sisällä WebView'n kaltaista teknologiaa. Tämän vuoksi hybridisovelluskehykset työpöytäkäyttöjärjestelmille käyttävät verkkoselainta, joka on riisuttu ylimääräisistä ominaisuuksista, samankaltaisen toiminnallisuuden saavuttaakseen. (4, 5)

Vaikka hybridisovellukset ovat pohjimmiltaan vain verkkosivuja, ne näyttävät päällepäin tavallisilta natiivisovelluksilta. Hybridisovelluskehykset tuovat mukanaan toimintoja, joita perinteisesti verkkosivulla ei ole. Näihin toimintoihin kuuluvat esimerkiksi kameran hallinta ja käyttöjärjestelmän natiivi-ilmoitukset.

Hybridisovelluskehyksien kuninkuudesta kamppailee tällä hetkellä kaksi ohjelmistokehystä: GitHubin kehittämä Electron ja vapaassa kehityksessä oleva NW.js. Service Suite Clientin alustaksi valintakriteereinä ovat hyvä ja jatkuva tuki, ominaisuuksien määrä ja käyttöjärjestelmätuki. Näistä kriteereistä NW.js vie itselleen ominaisuuksien määrän, mutta jää toiseksi kahdessa muussa kriteerissä, joten valinta kohdistuu Electroniin. (5, 6)

4.1 Electron

Electron on GitHubin kehittämä vapaan lähdekoodin ohjelmistokehys. Electronin avulla on mahdollista kehittää järjestelmäriippumattomia työpöytäsovelluksia käyttäen web-teknologioita kuten HTML, CSS ja JavaScript. Electron toteuttaa tämän yhdistämällä Node.js-kirjaston ja avoimen lähdekoodin Chromium-verkkoselaimen yhdeksi prosessiksi. Googlen kehittämä Chrome-verkkoselain pohjautuu Chromiumiin. Electronilla toteutettu sovellus voidaan paketoida yleisimmille käyttöjärjestelmille sopiviksi. (5)

Electronin kehittäminen aloitettiin vuonna 2013 Atom Shell -nimisenä projektina. Projektin alkuperäinen tarkoitus oli olla alustana GitHubin kehittämälle tekstieditori Atomille. Kuitenkin vuonna 2014 sekä Electronin että Atomin lähdekoodit julkaistiin avoimena. Electron sai sovelluskehittäjiltä lämpimän vastaanoton ja sen voi nähdä nykypäivänä hyvin Electroniin pohjautuvien sovellusten määränä. Electronin päälle rakennetaan nykyään yhä suurempia ja monimutkaisempia sovelluksia hyvinä esimerkkeinä pikaviestin Slack sekä GitKraken, graafinen käyttöliittymä Git-versionhallintaohjelmalle. (5)

4.2 Hybridisovellusten hyödyt ja haitat

Hybridisovellukset tuovat mukanaan paljon hyötyjä niiden kehittäjille, mutta sillä on hintansa. Perinteisesti natiivisovelluksia kehittäessä sovellus joudutaan ohjelmoimaan useaan kertaan. Jokaisella käyttöjärjestelmällä on omat ohjelmointikielensä, jotka kehittäjän pitää taitaa tukeakseen niitä jokaista. Hybridisovellusten käyttäessä yhtä ja samaa ohjelmointikieltä jokaisella käyttöjärjestelmällä on kehittäjän todella helppoa kehittää sovellus kaikille käyttöjärjestelmille toimivaksi. Pienen muutoksen tekeminen natiivisovellukseen vaatii sen tekemisen jokaiseen versioon erikseen, kun taas hybridisovelluksen kohdalla riittää, kun sen tekee vain kerran. Hybridisovelluksen päivittäminen voi olla käyttäjälle täysin näkymätön prosessi, toisin kuin natiivisovelluksen päivittäminen vaatii usein käyttäjältä toimenpiteitä. (7)

Käyttöjärjestelmäkohtaiset ohjelmointikielet ovat kuitenkin huomattavasti tehokkaampia ja resurssiviisaampia kuin hybridisovellusten ratkaisut. Jos sovelluksen tarkoituksena vaatii raskasta laskentaa, kuten grafiikan piirtämistä, ei sen toteuttaminen hybridisovelluksena ole järkevä ratkaisu. (7)

Vaikka hybridisovellukset on rakennettu saman selaimen, Chromiumin, päälle, käytännössä silti joudutaan jokaista sovellusta varten tuomaan oma versio samasta selaimesta, koska eri sovellukset on mahdollisesti rakennettu eri selainversion päälle. Hybridisovelluksessa sen mukana tuleva selain on usein suurin tiedosto sovelluksessa. Tämä ei ole loppupeleissä kovin järkevä tapa toimia, mutta tällä hetkellä se on välttämättömyys.

5 Magnetoenkefalografia

Magnetoenkefalografia, lyhennettynä MEG eli aivomagneettikäyrä, on aivojen mittaamiseen käytetty kuvantamismenetelmä. MEG:lla aivojen tuottamia magneettikenttiä voidaan mitata pään ulkopuolelta. Ihmisen aivot toimivat hermosolujen sähkökemiallisella viestinnällä, mikä aiheuttaa heikkoja sähkökenttiä, joiden luomat magneettikentät voidaan mitata. MEG:ssa näitä magneettikenttiä mitataan millisekunnin välein ja jopa muutaman millimetrin tarkkuudella. MEG:lla saatua dataa voidaan visualisoida ja siten saada tietoa siitä, millä aivoalueella tapahtuu paljon hermosoluviestintää, kun mitattava henkilö tekee erilaisia tehtäviä. (8, 9)

Magnetoenkefalografiaa käytetään esimerkiksi epilepsiapotilaiden hoidossa. Sen avulla voidaan paikantaa, millä aivoalueilla epilepsiapesäkkeet sijaitsevat. MEG:iä käytetään myös ennen aivoleikkauksia, toiminnallisesti tärkeiden aivoalueiden paikannukseen. (8, 9)

5.1 Elekta Oy

Tämän työn tilaaja Elekta Oy on suomalainen lääketieteen alalla toimiva yritys, jonka päätoimiala on suunnitella, valmistaa ja myydä neuromagneettisia mittauslaitteita. Elekta Oy on osa suurempaa ruotsalaista Elekta AB:tä, jonka osana on myös muita eri lääketieteen aloilla toimivia yrityksiä.

Elekta Oy:n tärkein tuote on Elekta Neuromag TRIUX -laite (kuva 3) ja siihen kuuluva lisäjärjestelmä Internal Helium Recycler, IHR. TRIUX on maailman johtava MEG-mittauslaite, ja IHR on siihen kuuluva järjestelmä, jonka käyttötarkoitus on kierrättää mittauslaitteen vaatimaa nestemäistä heliumia.

MEG-laitteet vaativat sekä jatkuvaa ennakoivaa huoltoa että korjaamista, joiden digitalisointi on tämän työn päämäärä.

Kuva 3. Elekta Neuromag TRIUX

5.2 MEG-työskentely-ympäristö

MEG-laitteet ovat yleensä asennettu joko sairaaloihin tai yliopistoihin. Yleensä MEG-laitteet sijaitsevat laboratorioissa, jotka on rakennettu maan pinnan alapuolelle. Laboratorioiden sisällä on magneettisesti suojattu huone, jonka sisälle itse laite on asennettu. Tämä on mittaustulosten kannalta hyvin tärkeää, koska aivojen aiheuttamat magneettikentät ovat todella heikkoja. (10)

Laboratoriossa suojahuoneen ulkopuolella sijaitsee data acquisition computer, DACQ, tietokone, jolla MEG-laitetta hallitaan ja jolla mittaukset suoritetaan (kuva 4).

Service Suiten sovelluksista kaksi, Service Suite DACQ ja Service Suite Client, pyörivät DACQ-tietokoneella.

Kuva 4. BioMag-laboratorio Helsingin yliopistollisessa keskussairaalassa

5.3 MEG-laitteen asennus ja huolto

MEG-laitteen asennukset ja huollot koostuvat kahdenlaisista työtehtävistä, manuaalisista mittauksista ja tietokoneella tehtävistä automaattisista mittauksista. Kummassakin tapauksessa mittaustulokset kirjataan käsin paperiseen huoltoraporttiin. Valmis huoltoraportti talletetaan turvakaappiin. Tällaisessa vanhanaikaisessa toimintamallissa on omat ongelmansa.

Huoltoa tehdessä on hankalaa verrata kyseisen kerran mittaustuloksia esimerkiksi edellisen vuoden mittauksiin, koska huoltoraportti on paperinen ja sijaitsee fyysisesti muualla. Mittausten vertailu suuremmassa skaalassa on suuren työn takana. Jos haluttaisiin esimerkiksi mitata tietyn mittauksen tuloksia viimeisen viiden vuoden aikana kymmenestä eri laitteesta, jouduttaisiin kaivamaan yhteensä 50 huoltoraporttia, joista etsiä tiedot.

Automaattisten mittausten eli testien tekeminen vaatii huoltomieheltä osaamista Linux-käyttöjärjestelmän käyttämisestä. Useassa tapauksessa huoltomiehellä ei kuitenkaan tarvittavia taitoja ole, ja testien ajamiseen tarvitaan erilliset ohjeet. Testien jakelu ympäri maailmaa sijaitseviin laboratorioihin on myös haasteellista, koska kaikissa laboratorioissa ei ole internetyhteyttä.

Muun muassa näiden ongelmien vuoksi kehitetään digitaalinen huoltotyökalu Service Suite.

6 Service Suite

Service Suite on MEG-laitteen huoltoon ja ylläpitoon tarkoitettu palvelu. Sen tarkoituksena on tuoda erilaiset huoltoon liittyvät komponentit yhden digitaalisen palvelun alle.

Service Suite koostuu neljästä eri sovelluksesta:

- Service Suite Client, lyh. Client
    - huoltomiehen käyttämä huoltosovellus
- Service Suite DACQ, lyh. DACQ
    - DACQ-tietokoneella ajettava sovellus
- Service Suite Cloud, lyh. Cloud
    - Service Suiten REST-rajapinta
- Service Suite Admin, lyh. Admin
    - Service Suiten hallintaan tarkoitettu web-sovellus.

6.1 Sovelluksen toiminta

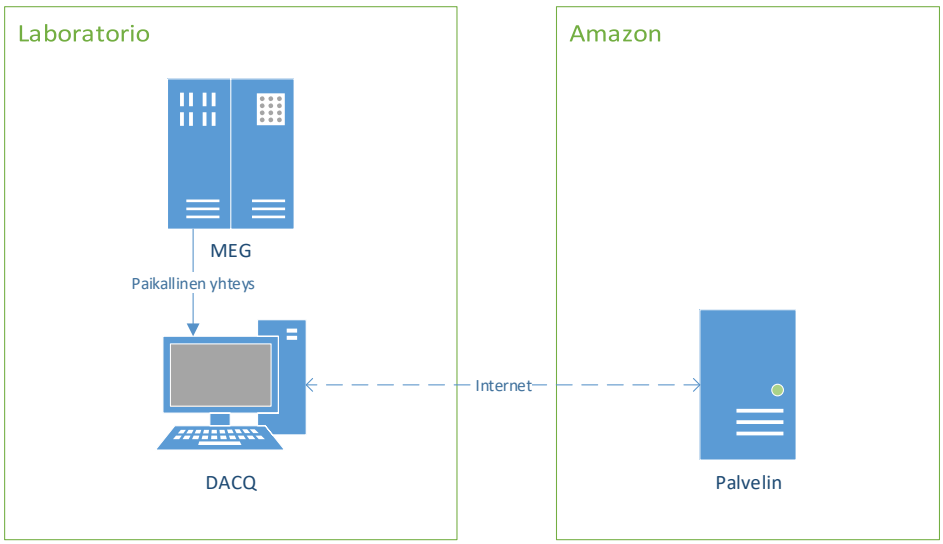
Service Suiten yleisin käyttötapaus on toimia alustana MEG-laitteen vuosihuollon raportoinnille ja toteutukselle. Huolto-ohjelmassa käyttäjälle avautuu lista tehtävistä, joista osa on manuaalisesti tehtäviä mittauksia laitteeseen ja osa automatisoituja skriptejä, testejä. Huollon yhteydessä käyttäjä voi ajaa erilaisia työkaluja, mutta ei yksittäisiä testejä. Manuaaliset työvaiheet ja testien tuottamat raportit kootaan yhdeksi

huoltoraportiksi, joka tallennetaan Service Suite Cloudiin. Käyttäjä voi huolto-ohjelman valmistuttua tarkastella huoltoraporttia sekä myös ajaa yksittäisiä testejä.

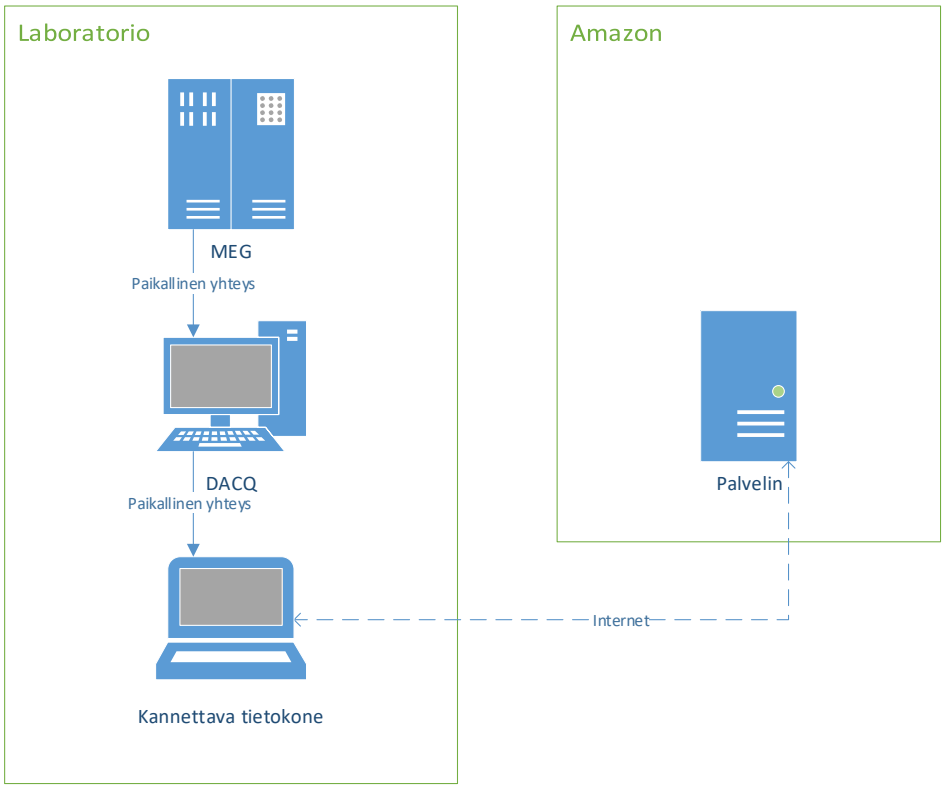
Testejä, työkaluja ja huolto-ohjelmia kutsutaan moduuleiksi. Moduuleihin tehdään aika ajoin päivityksiä, ja niiden päivittäminen tapahtuu Service Suiten avulla lataamalla ne Cloudista. Testit ja työkalut ovat käytännössä skriptitiedostoja, joita voitaisiin ajaa myös ilman Service Suitea. Vaikka skriptitiedostoja voi ajaa manuaalisesti. Se vaatii käyttäjältä osaamista Linux-käyttöjärjestelmästä. Testit pitävät sisällään raporttipohjan, joka täytetään testiajon tuloksella.

Service Suiten tapauksessa yllämainitut asiat eivät aina mene aivan näin suoraviivaisesti. DACQ-tietokoneista noin puolet on yhteydessä internetiin, jolloin yhteyttä Cloudiin ei ole. Näissä tapauksissa Service Suitea on silti voitava käyttää, joten sen pitää kyetä toimimaan ilman jatkuvaa internetyhteyttä.

Tilanteessa, jossa DACQ-tietokone on yhteydessä internetiin, Service Suite Client voidaan asentaa suoraan DACQ-tietokoneelle (ks. kuva 5). Jos taas internetyhteyttä ei ole, Service Suitea Clientiä voidaan käyttää huoltomiehen omalla kannettavalla tietokoneella (ks. kuva 6). Tällöin Client toimii väliaikaisena palvelimen korvaajana. Clientin mukana voidaan kuljettaa sekä päivitykset Cloudista DACQ-tietokoneelle asennettavaksi että raportit DACQ-tietokoneelta Cloudiin säilöttäväksi.



Kuva 5. Tilanne jossa DACQ-tietokoneella on internetyhteys

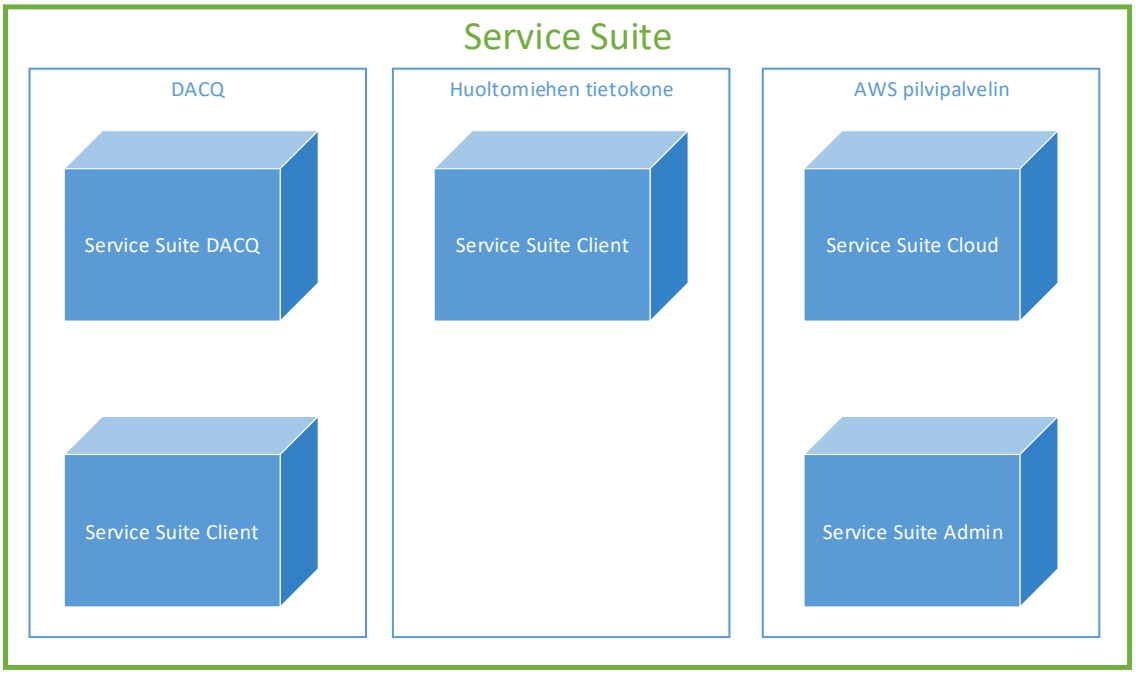


Kuva 6. Tilanne jossa DACQ-tietokoneella ei ole internetyhteyttä

Tallennettujen raporttien jälkeenpäin katselemiseen, moduulien uusien päivitysten julkaisemiseen ja käyttäjätietojen hallintaan on oma web-pohjainen sovellus, jonka käyttö on rajoitettu vain valituille henkilöille.

6.2 Service Suiten sovellukset

Service Suite Client on huoltomiehelle Service Suiten pääasiallinen käyttöliittymä. Client on sovellus, jolla huoltomies tekee huoltotoimenpiteet liittyen MEG-laitteeseen. Sillä voidaan päivittää moduuleja, käynnistää niitä ja siirtää niiden tuottamat raportit pilvipalvelimelle. Clientissä on myös mahdollisuus selailla ja katsella yhteydessä olevaan DACQ-tietokoneeseen tallennettuja raportteja. Client voidaan asentaa joko käyttäjän omalle kannettavalle tietokoneelle tai DACQ-tietokoneelle(ks. kuva 7).



Kuva 7. Service Suiten sovellukset

Service Suite DACQ on DACQ-tietokoneelle asennettava ohjelma (ks. kuva 7), jonka tehtävänä on olla yhteys Clientin ja MEG-laitteen välillä. DACQ ei ole koskaan suoraan yhteydessä Cloudiin, koska yhteyttä ei voida taata jokaisessa laboratoriossa.

DACQ:n kaksi tärkeintä ominaisuutta ovat moduulien skriptien ajaminen ja niiden tuottamien raporttien tallennus. Skriptien ajaminen on toteutettu Node.js:n child_process-moduulilla, joka on aliprosessien ajamiseen tarkoitettu moduuli.

DACQ on Node.js-palvelinsovellus, jonka toiminta perustuu REST-rajapintamalliin. Sovellus on kehitetty Express.js-kirjaston avulla ja tietokantayhteys Mongoose-kirjastolla. Tietokantana sovelluksella on DACQ-tietokoneelle asennettu MongoDB.

REST-rajapinta koostuu useista poluista, jotka ovat kaikki sisäänkirjautumisen takana. Rajapinnan polut paljastavat tietokannan CRUD (Create/luo, Read/lue, Update/päivitä, Delete/poista) -periaatteen mukaisesti. Tämä tarkoittaa sitä, että jos tietokannasta löytyy esimerkiksi testit-taulu, rajapinnasta löytyvät polut uuden testin luomiselle sekä olemassa olevan testin lukemiselle, päivittämiselle ja poistamiselle.

Service Suite Cloud on Service Suiten palvelinsovellus. Se pyörii Amazon Web Servicesin EC2-virtuaalipalvelimella. Virtuaalipalvelimia on alkuvaiheessa yksi, ja se

sijaitsee Frankfurtissa. Haluttaessa nopeammat yhteydet kauempaa maailmalta, AWS mahdollistaisi Cloudin asentamisen esimerkiksi Aasian tai Amerikkaan.

Cloudin toiminta perustuu REST-rajapintaan, joka on toteutettu Express.js:llä. Rajapinta toimii Clientin ja palvelimilla olevan tietokannan välillä. Tietokanta on MongoDB-tietokanta, jonka kanssa kommunikoimiseen käytetään Mongoose-kirjastoa.

Tietokantaan on tallennettuna testien ja huolto-ohjelmien tulokset, tiedot uusimmista testeistä ja huolto-ohjelmista. Tietokanta toimii myös käyttäjätietokantana. Käyttäjätietokantataulussa olevat tiedot ovat salattuja niiden arkaluontoisuuden vuoksi.

Rajapinnasta löytyy myös polut käyttäjän sisäänkirjautumista ja sisäänkirjautumistilan tarkastamiselle. Sisäänkirjautumisominaisuudet ovat toteutettu Passport.js-kirjastolla, joka tarjoaa useita erilaisia kirjautumismetodeja. Service Suiten tapauksessa metodina on sähköpostiosoitteen ja salasanan yhdistelmä.

Service Suite Admin on web-sovellus Service Suiten hallintaa varten. Siitä kerrotaan lisää luvussa 6.4.

6.3 Service Suite Client

Service Suite Client on kehitetty Electronilla, joten se on alustariippumaton ja se voidaan asentaa sekä DACQ-tietokoneelle että huoltomiehen omalle kannettavalle tietokoneelle. Sovellus on toteutettu modernina web-sovelluksena, joka pohjautuu AngularJS:ään ja sen palvelinlogiikka on toteutettu Express.js:llä.

Service Suite Clientin käyttöliittymän suunnittelun on tehnyt kaksi Metropolian kulttuurialan opiskelijaa.

Service Suite Client koostuu kuudesta päänäkymästä, jotka ovat

- kirjautuminen
- koti

- huolto-ohjelma
- testit
- työkalut
- raportit

Clientin reititys on toteutettu AngularJS:n ngRoute-luokan avulla. Reititys ei kuitenkaan ole käyttäjälle näkyvissä, koska Electron-sovelluksissa ei ole näkyvää osoitepalkkia. Näkymät ja niiden sisällä olevat ponnahdusikkunat ovat AngularJS-komponentteja.

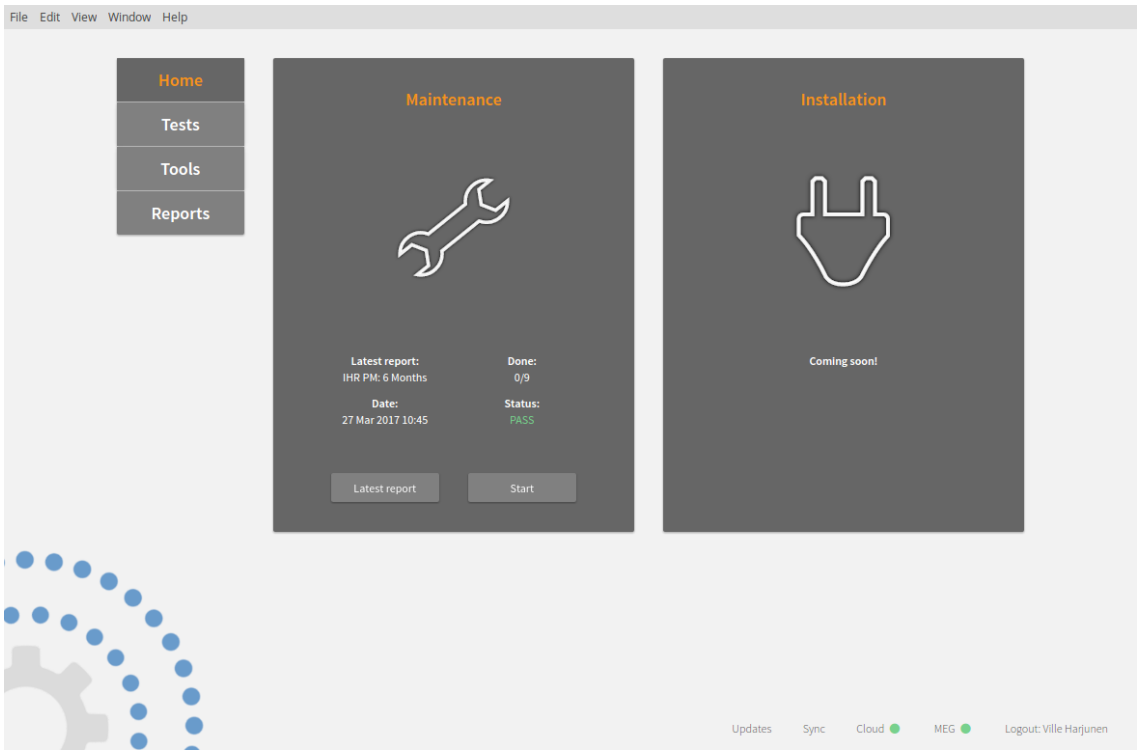
Sisäänkirjautuminen

Service Suite Client näkyy käyttäjälle työpöytäsovelluksena, jonka käynnistäessä käyttäjä päätyy sisäänkirjautumisnäkymään. Sisäänkirjautumisnäkymän käyttöliittymässä näkyy käyttäjätunnus- ja salasanakenttä sekä ikkunan alareunassa DACQ- ja Cloud-yhteyksien status. Yhteyksien status on nähtävissä sovelluksen jokaisessa näkymässä, koska yhteydet ovat sovelluksen käytön kannalta kriittisiä. Statukset on toteutettu reaaliaikaisilla WebSocket-yhteyksillä. Toisin kuin http-yhteydet, WebSocket-yhteydet ovat jatkuvia TCP-yhteyksiä, joten niistä voidaan päätellä yhteyden tila.

Service Suiten käyttäjätiedot on tallennettu Service Suite Cloudin tietokantaan. Käyttäjän syöttäessä sähköpostiosoitteensa ja salasanansa käyttäjätiedot lähetetään Cloudin REST-rajapintaan, joka vertaa niitä sen tietokantaan. Tämä ja kaikki muukin tietoliikenne Clientin ja Cloudin välillä on SSL-salattua. Jos sisäänkirjautumistiedot ovat oikein, käyttäjä ohjataan kotinäkymään. Käyttäjätiedot tallennetaan kryptattuina myös Clientin omaan tekstitiedostopohjaiseen tietokantaan, jotta käyttäjä voi myöhemmin kirjautua sisään sovellukseen tilanteessa, jossa Cloud-yhteyttä ei ole, eikä käyttäjätietojen varmistusta sieltä voida tehdä.

Koti

Kirjautumisnäkymän jälkeen käyttäjälle avautuu kotinäkymä (ks. kuva 8). Kotinäkymän ulkoasu riippuu siitä, onko Client yhteydessä Service Suite DACQ:n.

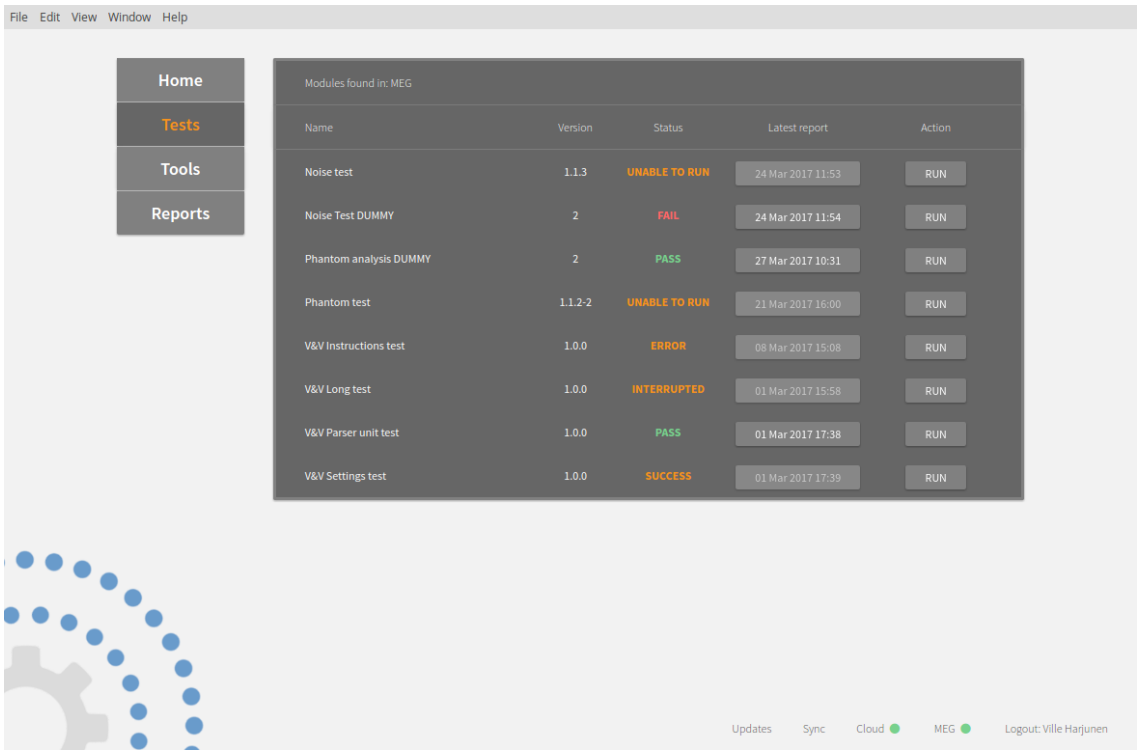


Kuva 8. Kotinäkymä

Tilanteessa, jossa DACQ-yhteys on olemassa, näkymän Maintenance-osiossa on sekä listattuna viimeisimmän tehdyn huolto-ohjelman tiedot ja mahdollisuus avata siihen liittyvä raportti että mahdollisuus aloittaa uusi huolto-ohjelma. Uusi huolto-ohjelma aloitetaan painamalla Start-painiketta, joka avaa ikkunan, johon on listattuna DACQ:lle asennetut huolto-ohjelmat. Valitsemalla jonkin näistä huolto-ohjelmista sovellus menee huoltomoodiin. Huoltomoodista ja raporteista kerrotaan enemmän myöhemmin. Jos yhteyttä DACQ:n ei ole, näkymän Maintenance ja Installation ovat himmennettyjä eikä niiden napeista tapahdu mitään.

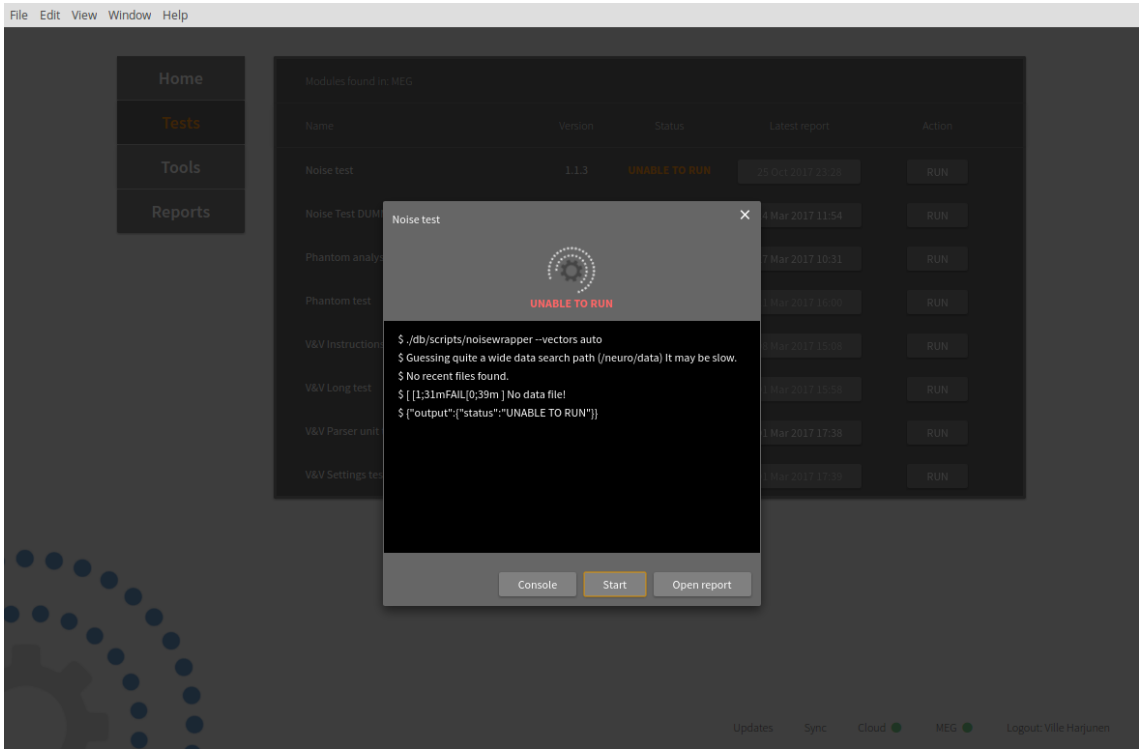
Testit ja työkalut

Sovelluksen Tests-näkymästä löytyy listaus testeistä (ks. kuva 9). Tools-näkymä näyttää lähes samalta kuin Tests-näkymä sillä poikkeuksella, että näkymässä ei ole Latest report -saraketta, koska työkalut eivät tuota raporttia. Testi- ja työkalulistaus listaa riippuen DACQ-yhteydestä joko Clientiin ladatut testit tai työkalut tai DACQ:n asennetut testit tai työkalut.



Kuva 9. Listaus testeistä

Näkymien RUN-painikkeet ovat himmennetyt ja poissa käytöstä silloin, kun Client ei ole yhteydessä DACQ:iin, koska niiden ajaminen ei onnistu ilman DACQ:ia. Tällöin listauksessa näkyvät versionumerot ovat Clientiin ladattujen moduulien versionumeroita. Tällöin myös listauksen Status- ja Latest report -sarakkeet ovat tyhjiä.



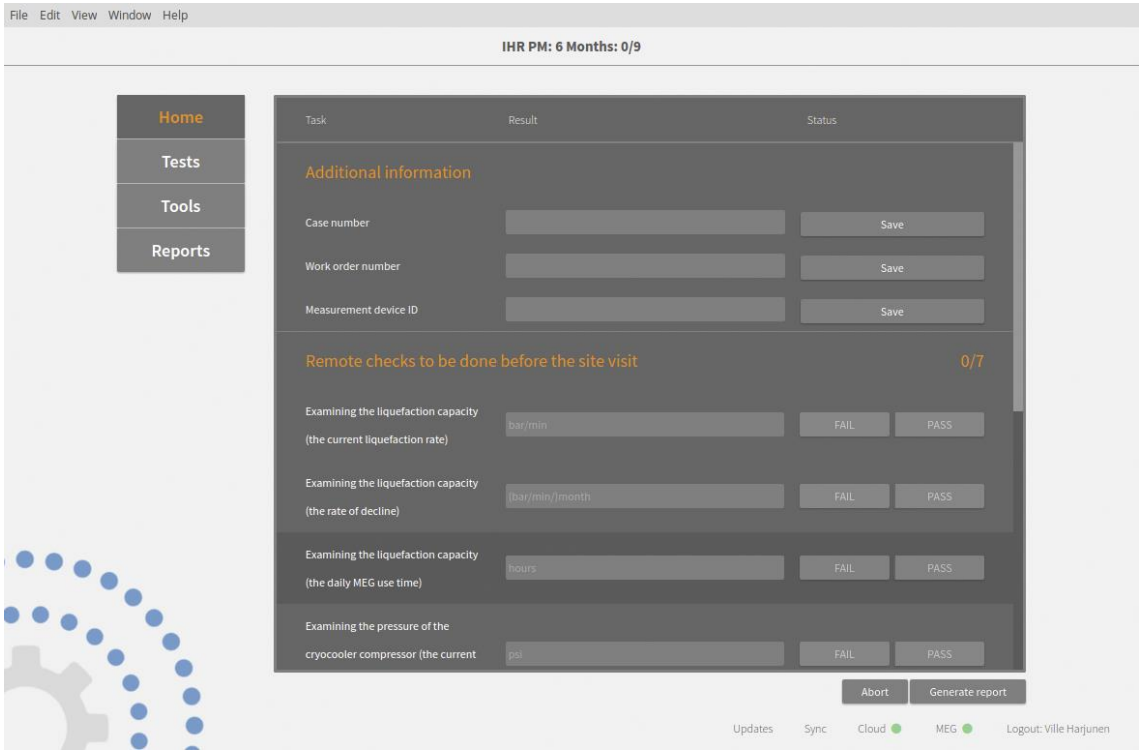
Kuva 10. Testin ajaminen

Tests- tai Tools-näkymässä käyttäjän painaessa jonkin moduulin RUN-painiketta käyttäjälle avautuu moduulinajoikkuna. Ikkunasta löytyvät kyseisen moduuliajon asetuslistaus ja Start-painike. Käyttäjän painaessa Start-painiketta moduulin ajo käynnistyy. Client lähettää DACQ:lle tiedon siitä, mitä moduulia ajetaan (ks. kuva 10). DACQ käynnistää kyseiseen moduulin liittyvän skriptitiedoston. Skriptit tuottavat ajon aikana komentoriville tekstiä, jota DACQ lähettää Clientille WebSocket-yhteydellä reaaliaikaisesti. Käyttäjä voi nähdä tämän tekstin painamalla Console-painiketta, joka avaa komentorivi-ikkunan näköisen ikkunan (ks. kuva 10). Jotkin skriptit vaativat kesken ajon käyttäjältä toimia, kuten esimerkiksi MEG-laite pitää kääntää makuuasennosta istuma-asentoon. Tällöin skripti jää odottamaan käyttäjän ilmoitusta siitä, että laite on käännetty. Skripti tulostaa ennalta määrätyn merkkijonon, jonka perusteella DACQ lähettää Clientille tiedon tästä. Käyttäjälle näkyy tässä esimerkkitapauksessa kehotus kääntää laite ja painike, jota painamalla käyttäjä ilmoittaa laitteen olevan käännetty. Painikkeiden lisäksi tuettuna on myös tekstisyöte. Moduulinajoikkunan voi sulkea ajon aikana, ja moduulin ajo jatkuu taustalla. Kun ajo on valmis, ikkuna pomppaa automaattisesti esiin.

Kun moduulin ajo on valmis, käyttäjälle näytetään testien kohdalla status ja työkalujen kohdalla vain tieto siitä, että ajo on valmis. Testien kohdalla käyttäjälle näytetään myös Open report -painike, jota painamalla käyttäjälle avautuu uuteen ikkunaan kyseisen testiajon tarkempi raportti, liite 2.

Huolto-ohjelmat

Service Suiten huolto-ohjelmat käynnistetään aiemmin kerrotulla tavalla. Huolto-ohjelmien tarkoituksena on niputtaa yhteen automaattisia testejä ja manuaalisia tehtäviä. Kun huolto-ohjelma aloitetaan, Client näyttää Koti-näkymän sijaan Huolto-näkymän (ks. kuva 11). Huolto-ohjelman ollessa päällä yksittäisiä testejä ei voi ajaa testinäkymästä. Työkalujen ajamista ei ole estetty huollon aikana, koska niitä voi tarvita kesken huollon. Huolto-ohjelman tila tallentuu DACQ:n tietokantaan ja huolto-ohjelman ollessa päällä jokainen Client, joka siihen yhdistyy, käynnistyy automaattisesti huoltotilaan.



Kuva 11. Huoltonäkymä

Manuaaliset tehtävät tuottavat aina hyväksytyn tai hylätyn tuloksen. Käyttäjä täyttää niitä kirjoittamalla selitekenttään esimerkiksi mittaustuloksen ja painaa FAIL- tai PASS-painiketta. Painikkeen painaminen tallentaa selitteen ja tuloksen. Nämä tiedot lähetetään

DACQ:n REST-rajapintaan, jossa ne tallennetaan DACQ:n tietokantaan osaksi kyseistä huoltoa.

Huolto-ohjelmat voivat pitää sisällään automaattisia testejä. Ne erottuvat käyttöliittymästä siten, että niissä on RUN-painike FAIL- ja PASS-painikkeiden sijaan, eikä niissä ole selitettä. Testien ajaminen ei eroa yksittäisten testien ajamisesti Ne toimivat kuten luvussa 6.3.3 on kerrottu. Huolto-ohjelman aikana ajettujen testien tulokset tallennetaan tietokantaan myös osaksi kyseistä huoltoa.

Kun huolto-ohjelman tehtävät on tehty, käyttäjä painaa Generate report -painiketta, joka lopettaa huolto-ohjelman. Käyttäjälle aukeaa ikkuna, jossa on kooste huollosta, ja painike, josta huoltoraportti aukeaa uuteen ikkunaan. Huoltoraportti tallennetaan DACQ:n tietokantaan ja riippuen Cloud-yhteydestä, joko Clientiin tai Cloudiin.

Raportit

Service Suite Clientissä käyttäjän avatessa raportin aukeaa uusi ikkuna, jossa raportti näytetään. Uuden ikkunan avaamiseen käytetään Electronin ali-ikkunaominaisuutta. Ali-ikkunalle lähetetään kyseisen tuloksen raporttipohja ja raporttidata. Raporttipohjat ovat HTML-muotoisia tiedostoja, jotka renderöidään raporttidatalla täydennettyinä. Raportti-ikkunan yläreunassa on painikkeet raportin tallentamiseen PDF-muodossa ja raportin tulostamiseen. Raportin tallennus PDF-tiedostoksi tapahtuu Electronin sisältämän Chromiumin verkkosivun tallentamiseen tarkoitetulla ominaisuudella. Liitteestä 1 löytyy PDF-muotoon tallennettu esimerkkiraportti.

Tämänhetkisten lainsäädännöllisten syiden vuoksi raportit joudutaan tulostamaan ja allekirjoittamaan käsin, jonka jälkeen paperiraportit arkistoidaan.

Moduulien päivitys

Kun käyttäjä saapuu kotinäkymään, sovellus suorittaa taustalla mahdollisten päivitysten tarkastuksen. Päivitysten tarkastus tapahtuu eri tavoin riippuen DACQ- ja Cloud-yhteyksistä. Optimitilanteessa, jossa molemmat yhteydet löytyvät, päivitysten tarkistus tehdään DACQ:n ja Cloudin välillä. Client pyytää DACQ:lta listauksen sen moduuleista ja lähettää ne Cloudin vertailtavaksi. Vertailun jälkeen Cloud palauttaa listan mahdollisista päivityksistä, ja käyttäjälle aukeaa ikkuna, jossa on listattu päivitettävät

moduulit. Käyttäjän painaessa Install updates -painiketta Client lataa päivitetyt moduulit Cloudista ja lähettää ne DACQ:lle. Käyttäjä voi myös olla päivittämättä moduuleja. Tällöin käyttöliittymässä olevan Updates-napin viereen ilmestyy ikoni muistuttamaan päivityksistä. Käyttäjä saa päivitysikkunan uudelleen avattua Updates-napista.

Moduulien tiedot tallennetaan DACQ-tietokoneella olevaan tietokantaan ja moduulien skriptitiedostot tallennetaan DACQ-tietokoneelle sovelluksen asennuskansioon.

Päivitykset voidaan myös asentaa Clientin kautta DACQ:n. Tällöin päivitysten on oltava entuudestaan ladattuna Clientiin. Päivitysten asennus Clientiin tapahtuu samoin kuin suoraan DACQ:n, mutta siten, että Client on yhteydessä Cloudiin ilman yhteyttä DACQ:n. Client pitää tietoa moduuleista sen omassa tietokannassaan ja tässä tapauksessa DACQ:n moduulien sijaan vertailu tapahtuu Clientin ja Cloudin moduulien välillä. Päivityksiä ladattaessa Clientin tietokantaan päivitetään päivittyneet tiedot moduuleista ja moduulien skriptitiedostot tallennetaan käyttäjän omalle kannettavalle tietokoneelle. Kun moduulit ovat ladattuina Clientiin ja Clientin, ja DACQ:n välinen yhteys on olemassa, päivitykset tarkastetaan Clientin ja DACQ:n moduulien välillä. Käyttäjän tietokoneelle tallennetut moduulien skriptitiedostot ovat tietoturvasyistä kryptattuja.

Raporttien synkronointi

Testit ja huolto-ohjelmat tuottavat raportteja, jotka halutaan tallentaa keskitetysti Cloudin tietokantaan. Käyttäjän ajaessa testejä tai huolto-ohjelmia niiden päätteeksi luodaan aina raportti. Tilanteessa, jossa on Cloud-yhteys olemassa, moduulin ajon jälkeen DACQ lähettää raportin Clientille, joka taas lähettää sen eteenpäin Cloudiin, joka tallentaa sen tietokantaansa, ja käyttäjä saa ilmoituksen synkronoinnin onnistumisesta.

Raporttien lähetys ei kuitenkaan onnistu tilanteessa, jossa laboratoriosta ei ole yhteyttä internetiin. Käyttäjän ajaessa testejä tai huolto-ohjelmia tällaisessa tilanteessa, niiden tuottamat raportit tallentuvat Clientin omaan tietokantaan. Kun Cloud-yhteys myöhemmin saavutetaan, käyttäjälle aukeaa automaattisesti sisäänkirjautumisen jälkeen synkronointi-ikkuna. Synkronointi-ikkunassa on listattu kaikki ne raportit, jotka Clientin tietokantaan on tallennettuna. Käyttäjän painaessa ikkunassa olevaa Synchronize-painiketta tallennetut raportit lähetetään Cloudiin, joka tallentaa ne omaan tietokantaansa. Kun Cloud saa raportit tallennettua, se ilmoittaa siitä Clientille, ja Client tyhjentää omassa tietokannassaan olevat raportit. Käyttäjä voi myös halutessaan jättää

synkronoinnin tekemättä. Tällöin käyttöliittymän Sync-napin viereen ilmestyy ikoni muistuttamaan siitä. Synkronointi-ikkunan saa avattua uudelleen Sync-napista.

6.4 Service Suite Admin

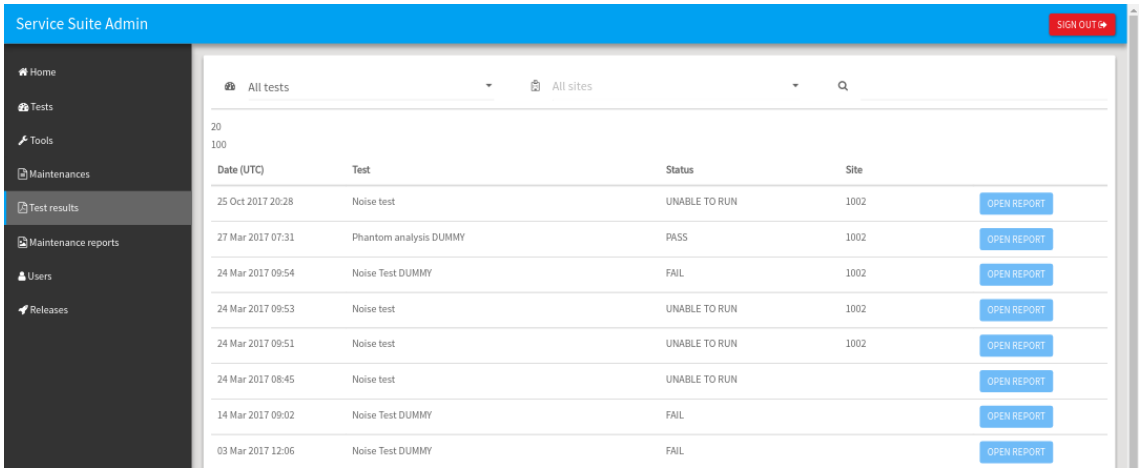
Service Suiteen liittyy monia sen hallintaan liittyviä asioita. Testit, työkalut ja huolto-ohjelmat päivittyvät tasaisin väliajoin ja uusia kehitetään jatkuvasti. Testit ja huolto-ohjelmat tuottavat raportteja, jotka ovat kootusti tallennettuna Service Suite Cloudin tietokantaan. Uusia käyttäjiä joudutaan luomaan ja unohtuneet salasanat on voitava nollata. Tämä tarkoitusta varten on Service Suite Admin, lyhyemmin Admin.

Service Suite Admin on moderni web-sovellus, joka on toteutettu MEAN-kirjastokokoelman työkaluilla. Adminin palvelinpuolen sovelluslogiikka pyörii samalla virtuaalipalvelimella kuin Service Suite Cloud ja käyttää samaa tietokantaa sen kanssa. Admin ei kuitenkaan käytä sisäänkirjautumista lukuun ottamatta Cloudin tarjoamaa REST-rajapintaa, koska rajapinta on toteutettu Clientin vaatimuksien mukaiseksi, joten se on Adminin näkökulmasta suurilta osin epäsopiva.

Kuva 12. Etusivu

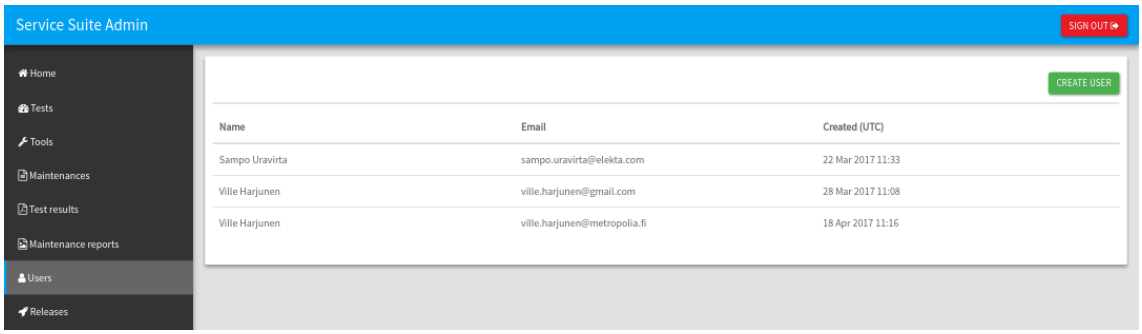
Adminin web-sovellus on toteutettu AngularJS:llä. Sisäänkirjautumisen jälkeen tulevalla etusivulla on kootusti kaikkien testien ja huolto-ohjelmien yleiskuva (ks. kuva 12).

Etusivulla on erilaisia graafeja, jotka kuvaavat muuan muassa eri testien onnistumisprosenttia.



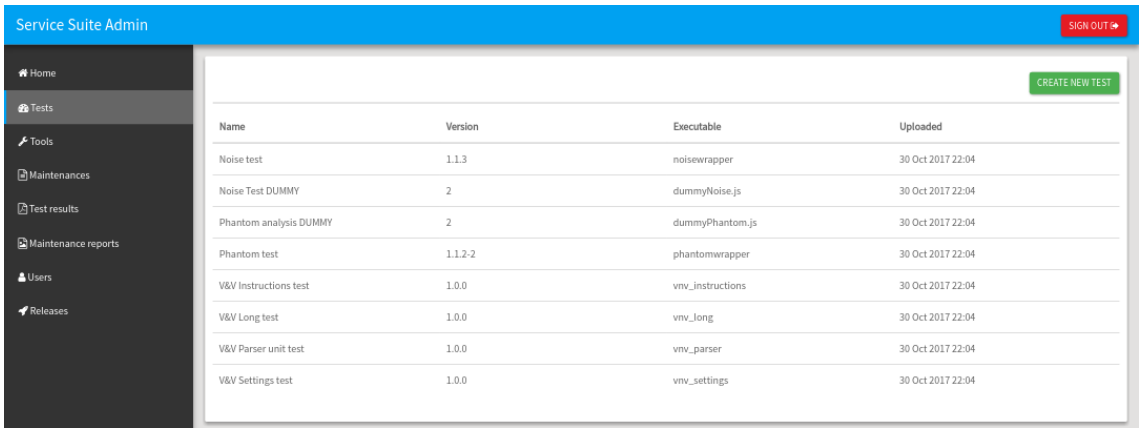
Kuva 13. Listaus testituloksista

Sovelluksesta löytyvät listaukset kaikista ajetuista testeistä ja huolto-ohjelmista. Listauksesta käy ilmi päivämäärä, käyttäjä ja se, oliko tulos hyväksytty vai ei. Tulosten raportit voidaan avata klikkaamalla OPEN REPORT -painiketta (ks. kuva 13). Raportit aukeavat samaan tapaan kuin Service Suite Clientissä.

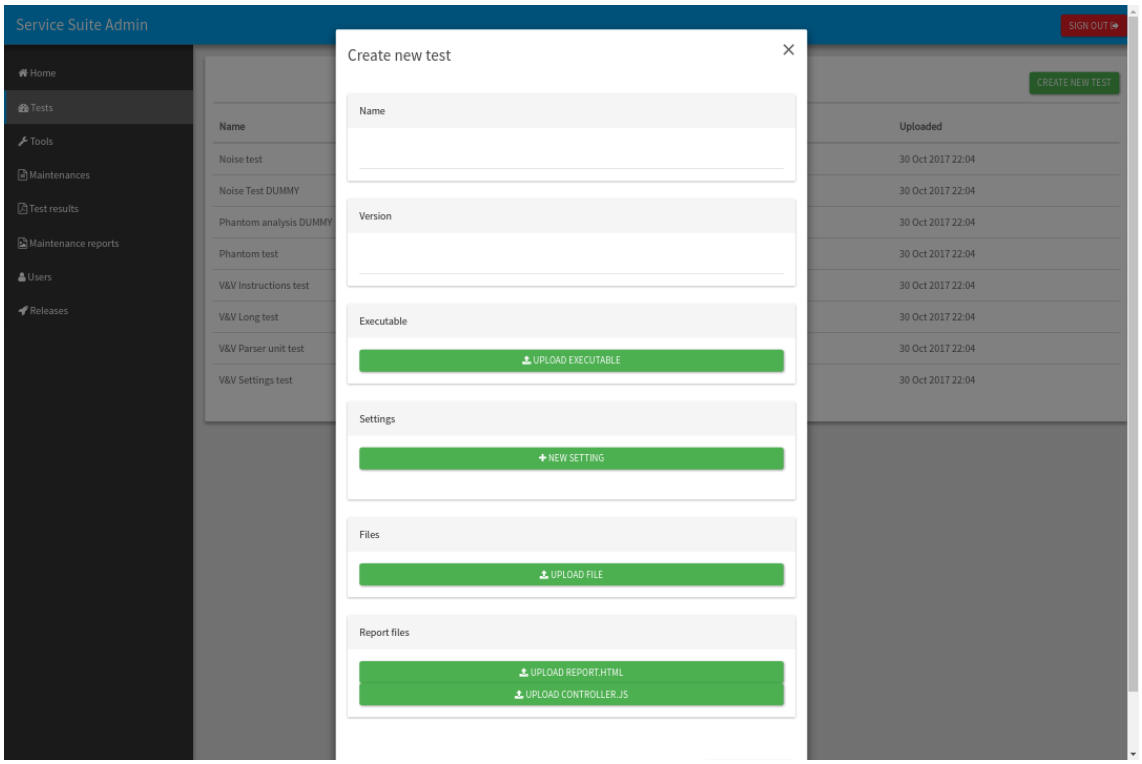


Kuva 14. Käyttäjienhallinta

Käyttäjienhallinnalla on oma näkymänsä (ks. kuva 14). Tässä näkymässä on listattu kaikki Service Suiten käyttäjät. Käyttäjiä voidaan lisätä painamalla CREATE USER-painiketta ja täyttämällä vaadittavat tiedot. Käyttäjien salasanat voidaan myös nollata tällä sivulla. Kun käyttäjän salasanan nollaa, käyttäjä saa sähköpostiinsa viestin, jossa on personoitu linkki sivulle, jossa voi asettaa itselleen uuden salasanan.



Kuva 15. Listaus testeistä

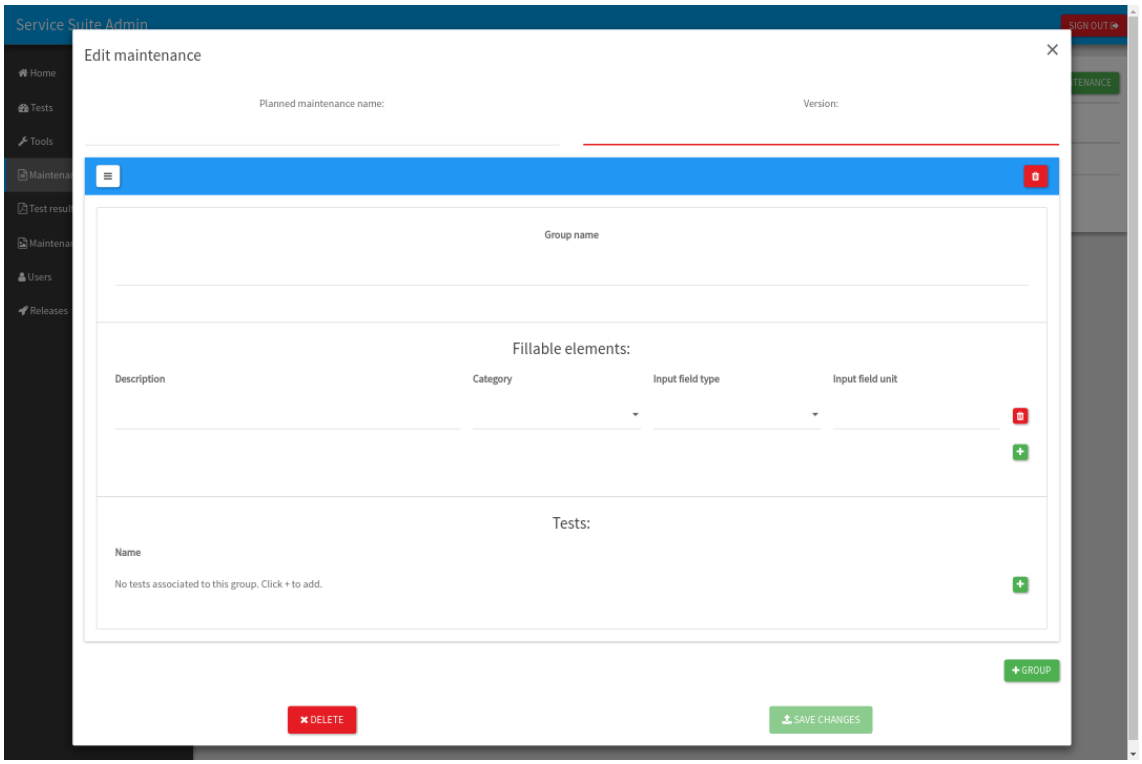


Kuva 16. Uuden testin luominen

Adminissa tapahtuu testien ja työkalujen sekä huolto-ohjelmien päivittäminen ja luominen. Testien ja työkalujen luominen ja päivitys ovat lähes samanlaisia prosesseja. Niissä erona on vain se, että testit vaativat raporttipohjan. Uusi testi tai työkalu lisätään painamalla CREATE NEW TEST/TOOL -painiketta (ks. kuva 15) ja täyttämällä kysytyt tiedot sekä valitsemalla skriptitiedoston ja testien tapauksessa raporttipohjan (kuva 16).

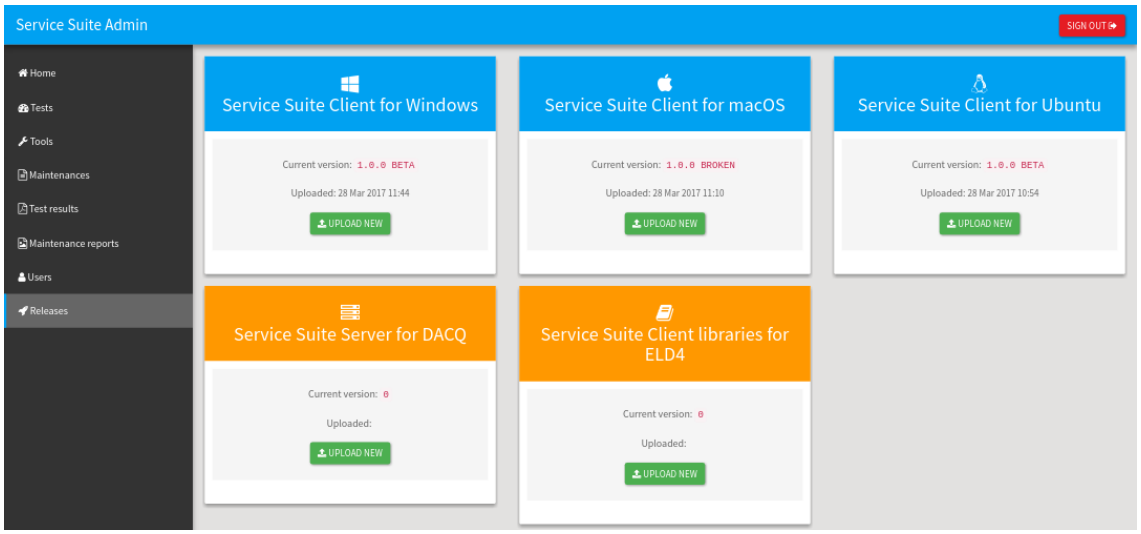
Testiriviä klikkaamalla aukeaa testin päivittämisikkuna. Päivittämisikkuna on samanlainen kuin uutta testiä luodessa, paitsi ikkunan kentät on valmiiksi täytetty nimen ja asetusten osalta.

Päivittämisen jälkeen kyseinen testi ilmestyy automaattisesti Service Suite Clientin päivitysikkunaan.



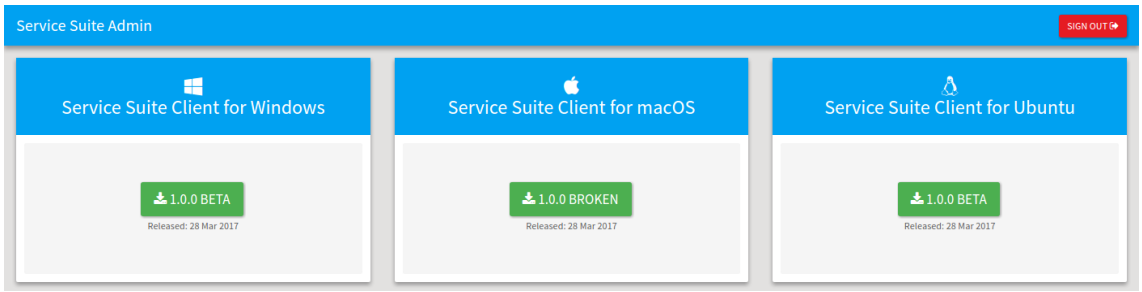
Kuva 17. Uuden huolto-ohjelman luominen

Huolto-ohjelmien luominen ja päivitys tapahtuvat lisäämällä, poistamalla ja muokkaamalla listattuja työtehtäviä (ks. kuva 17). Työtehtävät voivat olla joko automaattisia testejä tai käsin täytettäviä tekstikenttiä. Huolto-ohjelmaa päivittäessä päivitysikkunan kentät ovat testien päivitysikkunan tapaan valmiiksi täytettyjä. Luomisen tai päivittämisen jälkeen kyseinen huolto-ohjelma ilmestyy automaattisesti Service Suite Clientin päivitysikkunaan.



Kuva 18. Asennustiedostojen hallinta

Service Suite Clientin ja DACQ:n asennustiedostojen päivittämiselle on oma näkymänsä Adminissa (ks. kuva 18). Tämä näkymä näkyy vain valituille ylläpitäjäkäyttäjille. Klikkaamalla halutun asennustiedoston UPLOAD NEW -painiketta aukeaa ikkuna, jossa täytetään uuden version versionumero ja valitaan sen asennustiedosto.



Kuva 19. Asennustiedostojen lataussivu

Service Suite Clientin ja DACQ:n asennustiedostojen levitys tapahtuu Administa löytyvästä lataukset-sivulta (ks. kuva 19). Lataussivu näyttää automaattisesti kaikki ne asennustiedostot, joita kyseisellä hetkellä on ladattavissa.

7 Yhteenveto

Service Suiten toteutus onnistui hyvin. Kuten sovellusprojekteissa lähes aina, kehityksen aikana tuli ennalta arvaamattomia mutkia matkaan, mutta ne saatiin ratkaistua. Projektin alussa oli paljon suunnittelua ja sovelluksen käyttötarkoituksen kartoitusta, joka

myöhemmässä vaiheessa osoittautui hyödylliseksi. Useat monimutkaiset toteutukset oli jo alussa määritelty, joten niitä ei tarvinnut kesken kehityksen suunnitella.

Service Suite tehostaa asiakkaan huoltotoimintaa suuresti. Huolto-ohjelmien tekeminen nopeutuu ja suoraviivaistuu, sekä niiden raportit tallentuvat yhteen paikkaan, josta niitä on aiempaa helpompi vertailla.

Service Suite Clientin toteutus hybridisovelluksena oli järkevää. Sovellus ei vaadi juurikaan raskasta laskentaa tai grafiikan esittämistä, joten hybridisovelluksiin liittyvät haitat eivät tulleet esiin. Hybridisovelluksen kehittäminen vastasi nopeudeltaan lähes normaalin web-sovelluksen kehittämistä, joka on huomattavasti nopeampaa kuin natiivisovellusten toteuttaminen. Vaikka Service Suite ei ole perinteinen hybridisovellus, joka pohjautuu yhteen palvelimeen ja käyttöliittymään, sen toteutus hybridisovelluksena onnistui hyvin. Kuten Service Suiten tapauksessa hybridisovellusalustat taipuvat hyvin hieman monimutkaisempiinkin sovelluksiin, kunhan muistaa niiden rajoitukset.

Lähteet

1. What is MongoDB? https://www.mongodb.com/what-is-mongodb. Luettu 28.10.2017.

2. AngularJS dokumentaatio. https://docs.angularjs.org/misc/faq. Luettu 16.10.2017.

3. Juha Suomijoki: AngularJS Yksisivuisen web-sovelluksen käyttöliittymän toteutus AngularJS:llä. https://publications.theseus.fi/bitstream/handle/10024/90549/opinnaytetyo_juha_suomijoki.pdf?sequence=1. Luettu 28.10.2017.

4. Jeff Whatcott: HTML5 and the Rise of Hybrid Apps (2011). https://www.brightcove.com/en/blog/2011/11/html5-and-rise-hybrid-apps Luettu 13.8.2017.

5. Electron dokumentaatio. https://electron.atom.io/docs. Luettu 3.9.2017.

6. NW.js dokumentaatio. http://docs.nwjs.io/en/latest/. Luettu 29.10.2017.

7. 7 reasons to choose native application development. https://www.cleveroad.com/blog/7-reasons-to-choose-native-application-development-infographic. Luettu 29.10.2017.

8. Magnetoenkefalografia MEG. http://www.hus.fi/sairaanhoito/kuvantaminen-ja-fysiologia/tietoa-tutkimuksista/Magnetoenkefalografia-MEG/Sivut/default.aspx. Luettu 30.9.2017.

9. Magnetoenkefalografia. https://fi.wikipedia.org/wiki/Magnetoenkefalografia. Luettu 30.9.2017.

10. Miida Koponen, Heidi Kurkinen ja Emmi Vehola: MEG-tutkimuksen yhteydessä tehtävä EEG-elektrodien digitoinnin toistettavuus, 2015. http://www.theseus.fi/handle/10024/87541. Luettu 24.10.2017.

Esimerkkiraportti

Noise Test DUMMY

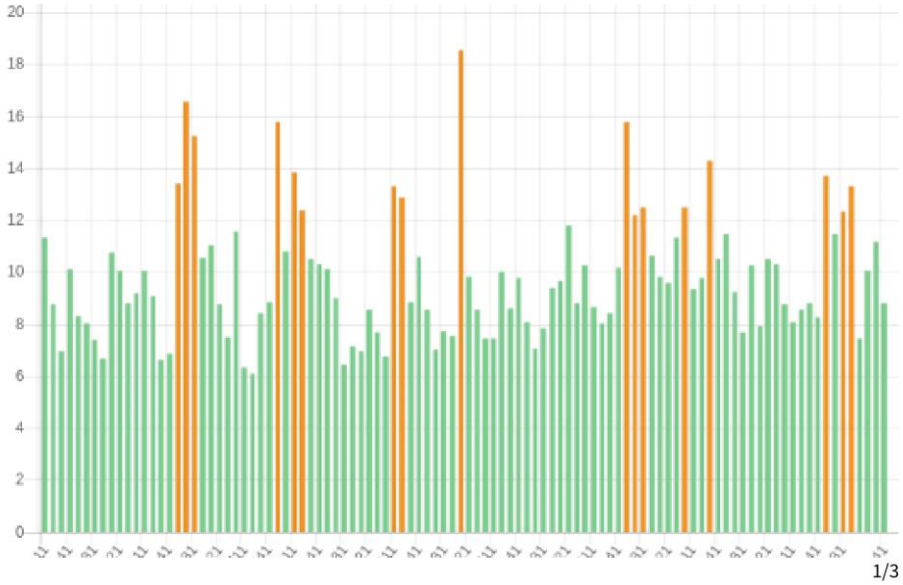
ELEKTA

24 Mar 2017 11:54

System number:
Site name:
User: Ville Harjunen
Test version: 2
IAS: Off
SSP Vectors: 6m 0g
File: /neuro/data/examples/service/biomag/emptyroom/emptyroom_supine_iasoff.fif

	Magnetometers (fT / √Hz)		Gradiometers (fT / cm / √Hz)	
	low frequency	white frequency	low frequency	white frequency
Mean noise	9.81	3.29	5.44	2.97
Number of good channels	85	102	204	201
Number of bad channels	0	0	0	0
Maximum noise	18.57	4.6	11.5	5.9
	FAIL	PASS	PASS	PASS

Magnetometers low frequency noise

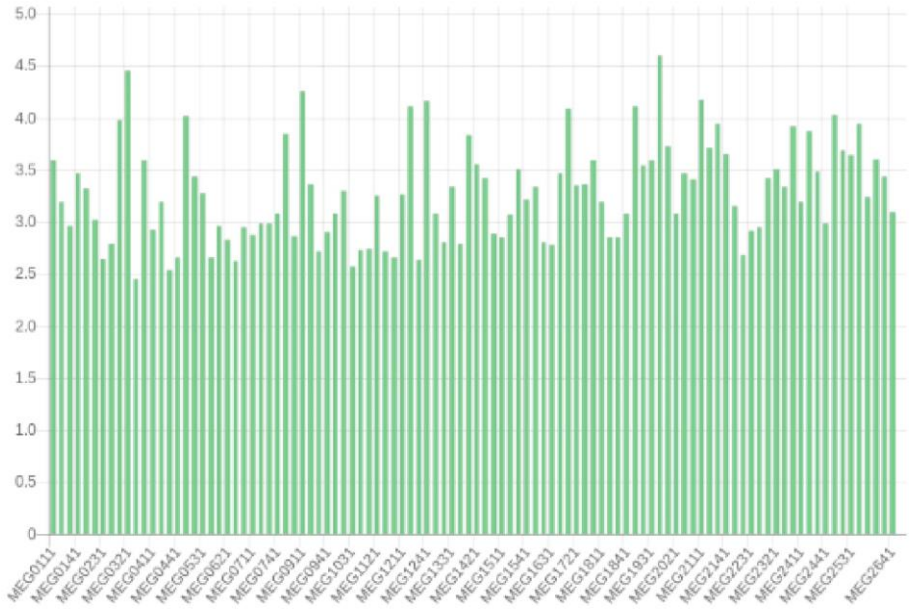


Noise Test DUMMY

ELEKTA

24 Mar 2017 11:54

Magnetometers white frequency noise

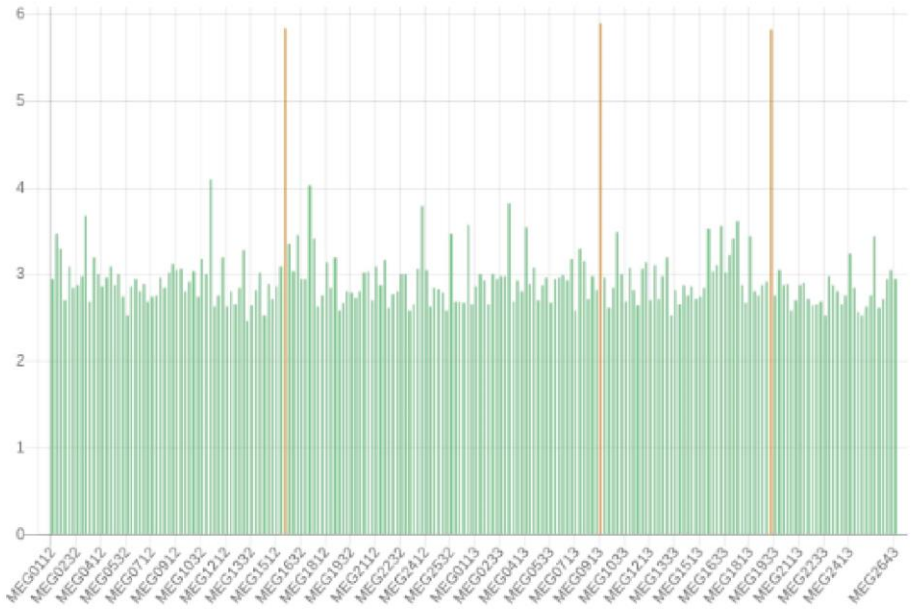


Gradiometers low frequency noise

Noise Test DUMMY

24 Mar 2017 11:54

Gradiometers white frequency noise



Comments

Signature and print name

Date