Tampereen ammattikorkeakoulu

Kuljettajaterminaalin tulevaisuus

Miska Hannunkivi

OPINNÄYTETYÖ
Joulukuu 2020

Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

HANNUNKIVI, MISKA:
Kuljettajaterminaalin tulevaisuus

Opinnäytetyö 32 sivua, joista liitteitä 1 sivu
Joulukuu 2020

Opinnäytetyössä tarkastellaan ohjelmistoyrityksen ympärivuorokautisessa käytössä olevan tuotteen osan vaihtoehtoisen ratkaisumallin mahdollisuutta. Työssä pohditaan mitkä haastavuudet pitäisi ottaa huomioon uuden ajan mukaisen toteutuksen kehitystyössä.

Työssä tutustutaan nykyiseen järjestelmään, jonka osana tämä toteutus toimii. Tarkennetaan sen ilmiannetut heikkoudet tuotannossa, jotka hankaloittavat kehitystyötä tarpeettomasti. Nykyisen järjestelmän tuntemus sen heikkouksineen toimii pohjana uuden toteutuksen tarpeiden määrittämisessä.

Työn edetessä esitetään huomioonotettavia ohjeita, jotka toimivat perustana uudelle toimintakonseptille. Näitä ohjeita sovelletaan tarvittavien toimintojen sekä tietojen käsittelyyn, jotka ovat annettu järjestelmän käyttöön, ylläpidettävyyden näkökulmasta.

Työn lopputulemana on joukko sääntöjä, mitä tarvitaan vakaaseen käyttökokemukseen uuden toimintamallin kehityksessä.

Asiasanat: kuljettajaterminaali, vaatimusmärittelyt, ohjelmistokehitys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information Technology
Software Engineering

HANNUNKIVI, MISKA:
The future of truck driver's terminal in material flow

Bachelor's thesis 32 pages, appendices 1 page
December 2020

The objective of this thesis was to examine the software requirements for an alternative solution for a product that is used by truck drivers around the clock. Research shows what should be taken care of in the development for a modern implementation.

At first the work introduces the current system, and how it is utilized in production. Then after understanding current system at adequate level, there is review about its reported weaknesses which unnecessarily complicates development and maintenance work. Gaining the knowledge about the current system with its flaws gives a solid ground to a beginning of the new solution.

Next a set of guidelines are introduced to give a base foundation for a new concept of operations. These guidelines are then applied to a consideration of required operations and data which system has access from the viewpoint of systems maintainability.

As a result, there are set of rules for providing what is needed to a solid user experience and what must be considered in a concept of a new solution.

Key words: truck driver's terminal, software requirements specification, software development

SISÄLLYS

LYHENTEET JA TERMIT

AJAX	Asynchronous JavaScript And XML, yhdistelmätekniikka vuorovaikutteiseen web-ohjelmistoon.
Apache HTTP Server	Avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma
Backend	Taustajärjestelmä, palvelinpuolen ohjelmistotekniikka.
CSS	Cascading Style Sheets, tyyliohje rakenteellisiin dokumentteihin
Frontend	Etujärjestelmä, käyttäjän päätelaitteelle suunnattu ohjelmiston osa.
HTML	Hypertext Markup Language, avoimesti standardoitu kuvauskieli
HTTP	Hypertext Transfer Protocol, tiedonsiirtoprotokolla
JavaScript	Juuri-ajallaan käännetty, dynaaminen ohjelmointikieli
Kehitysympäristö	Rajattu ympäristö, jota käytetään ohjelmiston kehitykseen.
LabView	Laboratory Virtual Instrument Engineering Workbench, National Instrumentsin kehittämä ohjelmistoympäristö.
Microsoft Windows	Microsoftin kehittämä graaffinen käyttöjäjestelmä
Netop	Etähallintaratkaisuita tarjoava ohjelmistoyritys.
Once by Pinja	Kaupallinen tuoteperhe, joka on kehitetty pitämään kirjaa materiaalivirroista.
ORM	Object-relational mapping, ohjelmointitekniikka muuntamaan data objektiksi kahden eriävän tyyppijärjestelmän välillä.
PC	Personal computer, yleisluontoinen mikrotietokone
PHP	Hypertext Preprocessor, ohjelmointikieli joka on kohdennettu palvelinympäristöjen dynaamisien web.sivujen luontiin.
SaaS	Software as service, ohjelmiston hankkiminen käytön laajuudesta maksettavana palveluna
Scrum	Viitekehys ketterän ohjelmistokehityksen projektinhallintaan

SQL	Structured Query Language, standardoitu kyselykieli relaatiotietokantoihin
TCP	Transmission Control Protocol, Tietoliikenneprotokolla tietokoneiden väliseen tiedonsiirtoon
Terminaali	Päätelaite, tietoliikenneyhteyden päässä sijaitseva laite, joka mahdollistaa kommunikoinnin loppukäyttäjän ja muun järjestelmän välillä.
T-SQL	Transact-SQL, perinteisen SQL-kielen muunnelma, jossa voidaan rakentaa myös toiminnallisuutta
Tuotanto	Ympäristö, jossa loppukäyttäjät käyttävät kehitettyä tuotetta.
UI	User Interface, käyttöliittymä
UX	User Experience, käyttäjäkokemus

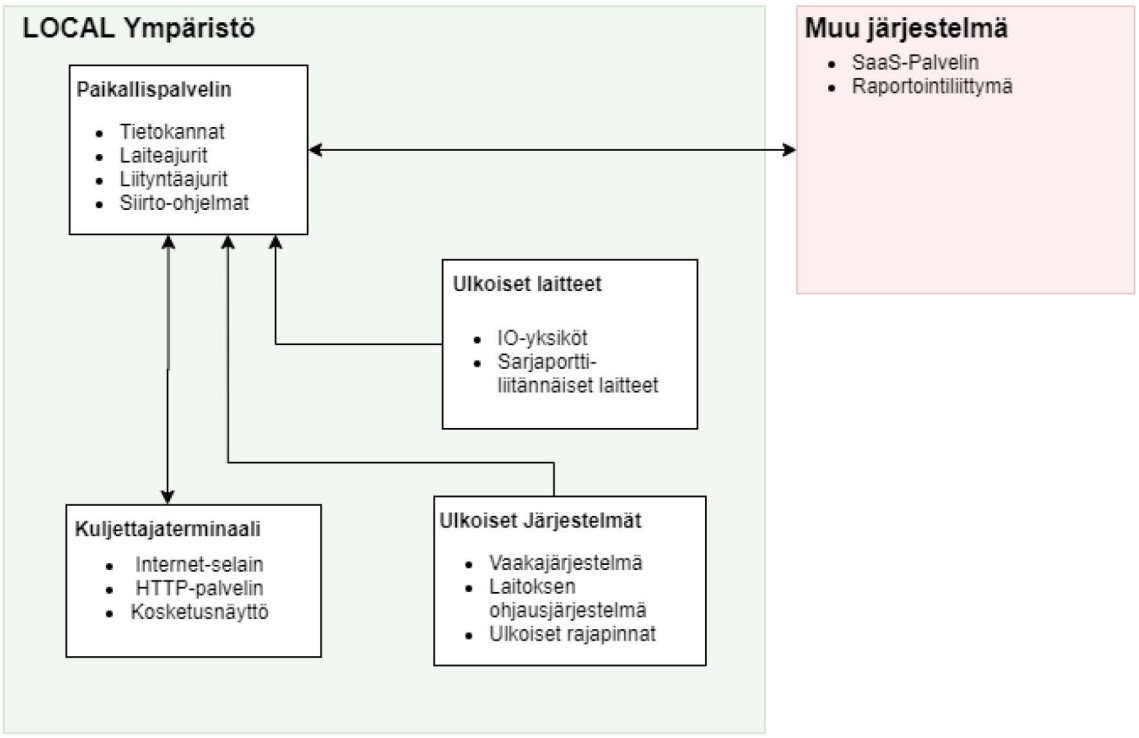
1 JOHDANTO

Opinnäytetyössä selvitetään korkean tason kuvaus vaatimuksista tulevaisuuden toteutuksille Once by Pinja -tuotteen kuljettajaterminaalin kokonaisuuteen, josta tarkemmin terminaalitietokoneella sijaitsevaan ohjelmistopinoon. Kuljettajaterminaali, jota työssä käsitellään, on tuotannossa toimiva fyysinen kokonaisuus, joka on tuotettu palvelemaan energiayhtiöiden käyttöpaikkojen tarpeita energiavirtojen kirjauksissa.

Työn toimeksiantajana toimii Pinja Digital Oy. Työ on rajattu toimeksiantajan Once by Pinja -tuotteen kuljettajaterminaalin ja sen tarjoamaan palvelukerrokseen. Kuljettajaterminaalin hyödyt tuotannon tasolla ovat selkeät, ja tässä työssä esitellään sen vahvuudet sekä heikkoudet ja pohditaan miten tulevaisuudessa olisi mahdollista järjestää vastaavanlainen palvelukerros energiavirtojen kuljetusvaiheen siihen kohtaan, missä tehdään kuorman kirjaus laitokselle, kuorman purku sekä lähtökirjaus päivittäen kirjattua kuormaa, joka luotiin laitokselle saavuttaessa. Työssä keskitytään kyseiseen palvelukerrokseen ja laajuutena pidetään kaikki saavutettavuuteen liittyvät komponentit, jotka ovat kaikessa yksinkertaisuudessaan elementit, jotka ovat pakollisia kuorman kirjauksen paikalliseen toimintaan.

2 JÄRJESTELMÄN KUVAUS

Once on materiaalivirtojen hallintaan keskittynyt selainpohjainen, SaaS-palvelumallin järjestelmä, jota hyödynnetään pääosin energiantuotantolaitosten ja kiertotalousalalla toimivien yritysten tilaus- ja toimitusketjujen hallintaan.



KUVA 1. Once Local -ympäristön yleiskuvaus

Materiaalivirtojen hallintajärjestelmätuote ei rajoitu työssä käytävään osaan, joka sisältää vain paikallisen, käyttöpaikoille sidotun käyttöliittymän materiaalikuormien kuljettajille. On oleellista tietää, että työssä käsiteltävä osa on vain optio tuotteen myynnin kannalta. Tuotteeseen kuuluu useampia ratkaisumalleja erilaisiin tarpeisiin, mitä energiavirtojen materiaalihallintaan tulee. Tuotteen kuljettajaterminaali on mahdollisesti jopa käytetyin komponentti materiaalivirtojen hallinnassa mikrotasolla.

2.1 Järjestelmän teknologiat

Järjestelmän tiedon lähteenä toimii relaatiotietokantainstanssi, jossa data varastoidaan sekä käsitellään tietoa sisältävien taulujen relaatioilla. Tätä tietokan-

tainstanssia tarjoillaan sekä ylläpidetään Microsoftin SQL Server -tietokantahallintajärjestelmällä, jossa kyselyt tapahtuvat Transact-SQL kielellä. Tämä järjestelmä tukee proseduureja, funktioita sekä liipaisimia. Proseduurit sekä funktiot ovat kutsuttavissa tietokantainstanssin ulkopuolelta. Näiden kahden erona on se, että funktiot eivät itsessään voi muokata taulujen sisältöä niin kuin proseduurit. Liipaisimet ovat sidoksissa yksittäisiin tauluihin tehtyihin muutoksiin, ja niitä kutsutaan täten haluttujen taulumuutoksien kohdalla.

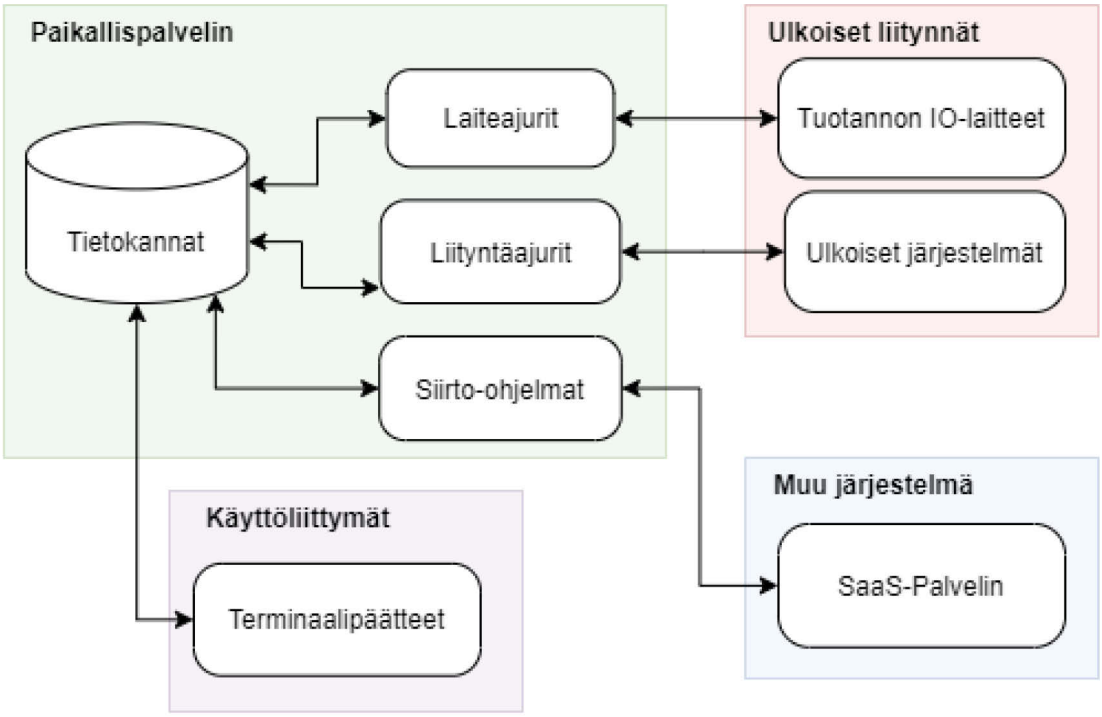
Tuotteen ydin on tietokanta, mutta järjestelmästä näkyvät osuudet ovat toteutettu frontend- että backend-järjestelmillä. Frontend-puolella toteutuksessa käytetään JavaScript, HTML ja CSS yhdistelmää rakentamaan ulkoasu toiminnallisuudelle, kun taas backend puolella tästä toiminnallisuudesta vastaa PHP-ohjelmointikieli. Järjestelmä tarvitsee myös HTTP-palvelimen yhteyksille, jotka ovat toteutettu Apache HTTP serverillä.

Järjestelmän juoksevia toimintoja suorittavat ajurit ovat ohjelmoitu LabView-ohjelmointiympäristöllä, jossa käytetään graafista-ohjelmointikieltä ja josta käännetään ajettava binääri.

2.2 Järjestelmän tekninen kuvaus

Tuotannon konfiguraatioista on olemassa yksinkertaisia, vakiohintaisia ratkaisuja, mutta todella usein asiakas tarvitsee tuotteen tarjonnasta useampia, lisäkehitystä vaativia, asiakkaan ympäristöön yksilöllisiä ratkaisuja. Nämä yksilölliset ratkaisut ovat kehitetty niin, että kehitystyö ei vaikuta valmiiseen pohjaratkaisumalliin vaan on täydennystä, eikä järjestelmän keskeistä toimivaa osaa.

Laitokselle asennettava järjestelmä on tarkoitettu ympärivuorokautiseen ajoon. Se tarjoaa kosketuskäyttöliittymät käyttäjille, hallitsee laiteajureita, sekä ylläpitää paikallisia rajapintoja. Kuljettajaterminaali sijaitsee järjestelmässä tuotteen alimmalla tasolla, jossa sitä hyödynnetään tiedon keräämiseen. Kuljettajaterminaalia käyttävät osapuolet koostuvat sen ostaneista asiakkaista ja heidän toimittajistaan. Terminaalit asennetaan asiakkaan tuotantotilojen läheisyyteen fyysisesti ja verkkoteknisesti, jonka ansiosta pystytään hyödyntämään saatavilla olevia liityntöjä tuotannossa.



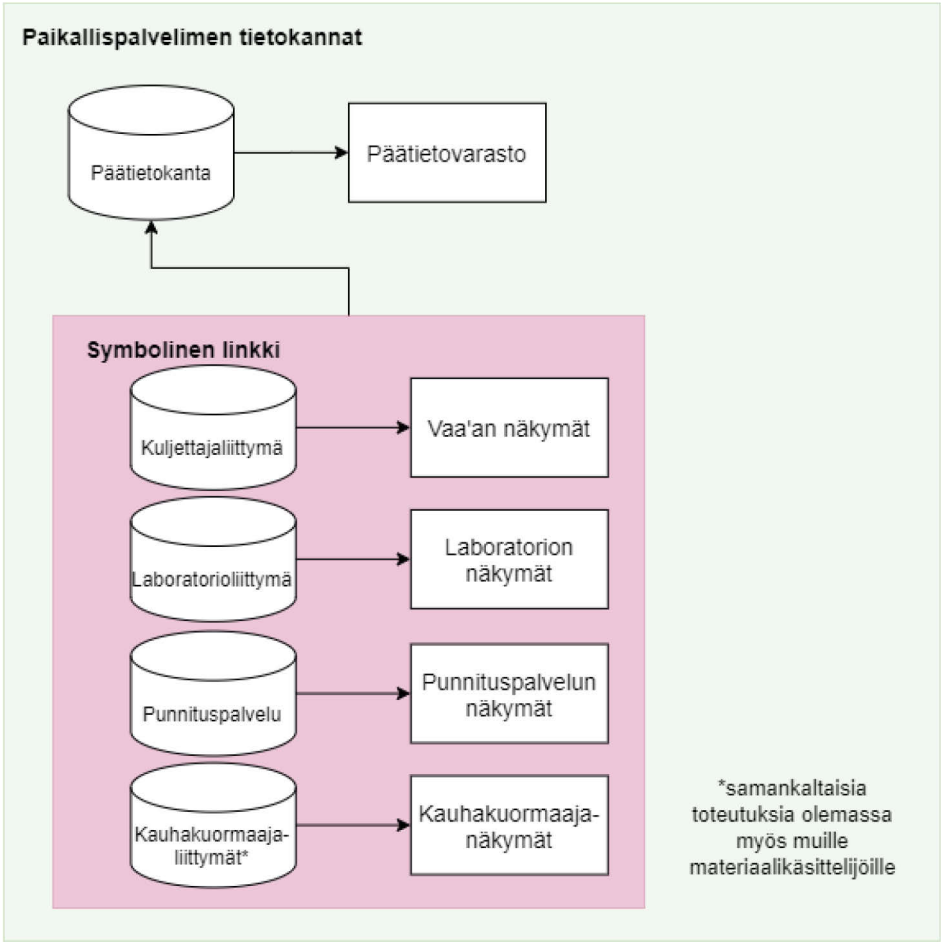
KUVA 2. Tuotannon järjestelmäkonfiguraatio

Ohjelmalogiikka on vahvasti sidottu tietokantoihin, joista saatava tieto ohjaa näkymiä terminaaleilla. Järjestelmän muut osat, jotka sisältyvät SaaS-palvelimeen, ovat materiaalivirtojen raportointi sekä hallintaliittymät. Uudet käyttöpaikat konfiguroidaan asiakkaille ensin raportointiin näkyviksi instansseiksi, joista sitten voidaan luoda oma paikallinen järjestelmä, uuden käyttöpaikan luonnista generoitujen, yksilöllisien tunnisteiden mukaan.

Tietokantayhteydet ovat järjestetty Apache HTTP palvelimen kautta kommunikoitavaan PHP-backendiin, johon muodostetaan yhteys internet selaimella. Frontend muodostuu pohjatoteutetuista JavaScript-komponenteista, jotka luodaan tietokannasta tulevan tiedon mukaisesti, sekä saavat täten myös toiminnallisuutensa. Tietokantakyselyiksi toteutetut komennot frontend komponenteissä kulkevat kevyen, PHP-ohjelmointikielellä toteutetun backendin kautta, jota frontendin JavaScript kutsuu AJAX-toimintojen avulla. Kuljettajaterminaalin kommunikaatio on täten kaksisuuntaista sekä pyritty järjestämään niin, että asiakkaan tarpeen mukaan, terminaali esittää tarpeen vaatiessa aktiivisesti päivittyviä tietoja näyttöpäätteellä.

Järjestelmän kommunikointi ulkoisten laitteiden sekä järjestelmien kanssa on toteutettu LabView-ajureilla, jotka ottavat yhteyden TCP- tai sarjaporttiliikenteellä. Ajurit ovat konfiguroitu ottamaan yhteys päätietokantaan. Tällä tietokantayhteydellä ajurit voivat lukea liityntöjen parametrit, sekä kirjoittaa laitteiden aktiivista tilaa tietokantaan. Tämäntapainen yhteys tietokantaan, joka sisältää yhteysparametrit sekä datalle tarkoitetut kentät, luo mahdollisuuden konfiguroida yhteyksiä dynaamisesti.

Laitoksessa sijaitsevan järjestelmän paikallinen palvelin sisältää aina vähintään päätietokannan, joka sisältää tarpeelliset taulut, näkymät sekä triggerit järjestelmän synkroniselle toiminnalle, yhteysohjeet ajureille sekä muuhun järjestelmään, tarpeelliset siirtokentät, kun tietoa synkronoidaan paikallisen- sekä SaaS-palvelimen kesken. Riippuen asiakkaan tarpeista, tietokantoja paikallisesti voi olla useita. Nämä tietokannat jakautuvat myydyn funktionaalisuuden mukaan. Pääpiirteittäin tietokannat saadaan jaettua pää-, kuljettajakäyttöliittymä-, laboratoriokäyttöliittymä-, punnituspalvelu- sekä kauhakuormaajankantoihin.



KUVA 3. Tietokantojen lajittelu

3 KULJETTAJATERMINAALI

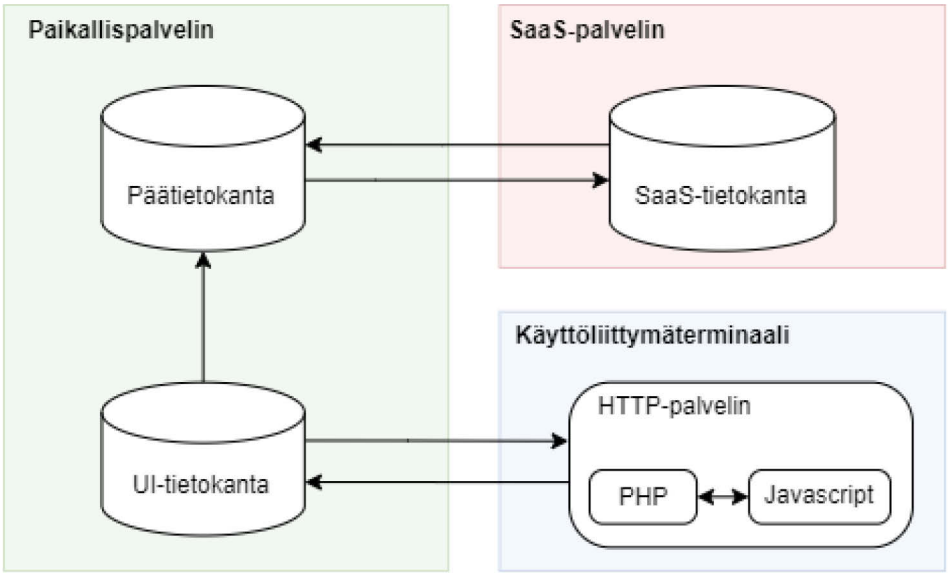
Paikalliseen kuormien kirjaukseen luotu terminaalikokonaisuus on tehty palvelemaan pääsääntöisesti energiakuljetuksien tekijöitä, jotka toiminnallaan terminaalipäätteellä luovat kuormatapahtumia laitoksille käyttöpaikalle ajetuista kuormista (LIITE 1). Tyypillinen kuljettajaterminaali on sijoitettu autovaa'an läheisyyteen niin, että ajoneuvon kuljettaja voi ajoneuvon avomesta ikkunasta käsin syöttää tarpeelliset tiedot kuorman kirjaukseen liittyen, poistumatta ajoneuvon kyydistä. Kun kuorma on saatu toimitettua käyttöpaikalle, kuljettaja päättää kuljetuksen kuljettajaterminaalin avulla päivittämällä järjestelmään ajoneuvon tyhjäpainon kuorman tietoihin, myös tarkastaen tapahtuman oikeellisuuden. Kuljettajaterminaalikokonaisuus sisältää kuormatapahtuman luonnin lisäksi mahdollisuuden tarjota käyttäjälle laitoskohtaista dataa visuaalisin elementein, kuin myös käyttöpaikan laiteohjauksia suorittavia elementtejä ruudulla oleviksi painikkeiksi muodostettuna.

Nykyterminaalin rakenteessa on tavoiteltu mahdollisimman suurta konfiguroitavuutta huomioiden asiakkaiden hyvin erinäköiset tarpeet. Toteutuksessa rakenne on sitten pirstaloitunut melko paljon, kun tuotteen kehityksen parissa on toiminut todella moni kehittäjä, joista jokaisella on ollut hieman erilaista osaamista sekä omanlaistaan näkemystä toteutusmalleihin.

Terminaalisovellus sisältää tuen monikielisyyteen, joka ratkaisee paljon käytännön haasteita laitoskohtaisten liikennöitsijöiden tapahtumista tuotannossa, jossa työvoima sisältää muitakin kuin äidinkieleltään suomenkielisiä henkilöitä.

3.1 Käyttöliittymäterminaalin tekninen toteutus

Käyttöliittymäterminaali on laitetasolla toteutettu kosketusnäytöllä varustetulla paneeli-PC:llä, jossa on käytössä Microsoftin kehittämä Windows käyttöjärjestelmä. Paneeli-PC:n minimivaatimukset ovat internet-yhteys, sekä tarpeellinen tehokkuus ajamaan graafista käyttöliittymää sulavasti internet-selaimella. Sulava käyttökokemus voidaan tulkita siten, että terminaalia käytettäessä käyttäjän ei tarvitse odottaa tiedonsiirrosta aiheutuvaa aikaväliä seuraavan komennon suorittamista varten, joka olisi merkitsevä tuotannon toiminnan suhteen.



KUVA 4. Käyttöliittymän tietokantayhteydet

Paneeli-PC:n on tarkoitus ylläpitää käyttöliittymäterminaalia, jota ajetaan internet selaimella kioskitilassa (koko ruudun peittävä, ulkoisilta palveluilta häiriötön tila), jossa internet-selaimen toiminnallisuus on rajoitettu vain tuotteen piiriin kuuluvalle käyttöliittymäterminaali-näkymälle. Tämä näkymä koostuu JavaScript -komponenteista, jotka ovat ennalta määrättyjä, mutta vaativat tietokannasta toiminnallisuuden. Tietokantayhteys kuljettajaterminaalille suoritetaan PHP-backend palvelun kautta, jota ajetaan Apache HTTP-serverillä. UI-tietokanta eroaa päätietokannasta juuri sillä osalla, joka määrittelee, mitä käyttöliittymä tarjoaa käyttökokemuspuolella. Tietoa ei kahdenneta tietokantojen välillä, vaan tietokannat käyttävät synonyymejä toistensa sisältöihinsä.

3.2 Havaitut haasteet kuljettajaterminaalin toteutuksessa

Työn tarkoituksena on pyrkiä löytämään nykyisiin pitkäaikaisiin, tuotteen palvelutasoa kuormittaviin tekijöihin ratkaisua edistävää pohdintaa niin asiakaspalvelun että kehityksen kannalta. Tuotteen kriittisimmät painopisteet ovat saatavuus sekä luotettavuus (tiedon eheys sisältyy luotettavuuteen). Saatavuuden maksimoimiseksi tuotteella on jatkuvan palvelemisen periaate, eikä tämä saisi olla luotettavuudelle este. Asiakkaalle myytävä vakiokokoonpano tuotantoon on asiakkaalle kuormien kirjauksen lisäksi oiva väline tuoda kuljettajille lisätietoa käyt-

töpaikkaan linkitetyistä tapahtumista, ohjata tietynlaisia kuormia ilman että luodaan kovakoodattuja estoja tunnisteelle sekä mahdollistaa vahvistettu tapahtuma kirjauksesta tunnistein ja salasanoin.

Nykyisellään on olemassa myös toteutuksia punnituksiin, missä todellisuudessa koko logiikka on sidottu erilaisiin tunnistimiin kentällä ilman graafista käyttöliittymää, mutta asiakkaalle mieluisin vaihtoehto on ylläpitää dynaamisuus kuormien kirjauksen suhteen.

Tuotteen vahvuuksiin kuuluvat pienet tai olemattomat katkokset tuotantoon konfiguroitaessa sekvenssiä määrääviä parametrejä, mutta ohjelmistokehittäjän näkökulmasta tuotteen ylläpidettävyys on heikkoa. Itse toiminnallisuuteen tuotannossa ongelmat eivät ylety, kun ei ole kyse monimutkaisesta toteutuksesta, mutta sen huono taso tekee kehittämisestä tarpeettoman haastavaa.

Käyttäjien ongelma voi olla yksinomaan siinä, että saapuessaan laitokselle kuljettaja ei ole varma kuljetettavastansa materiaalista tai mistä kuorma on peräisin. Tämä luo haasteen käyttöliittymälle sekä kuljettajalle, koska mahdollisesti huonosti tai liian vaikeasti toteutettu käyttöliittymä johtaa tietosisältövirheisiin, joiden korjaaminen maksaa.

Kun kehittäjä saa työkseen luoda tuotantoon uuden toteutuksen kuljettajaterminaalista, tätä ennen ovat projektin vetäjä sekä tuotteen myyntihenkilö asiakkaan kanssa käyneet lävitse halutun toiminnallisuuden tuotteen osalta. Myyty konfiguraatio on usein erittäin laaja-alainen sekä perustuu vahvasti aiempiin tuotannon toteutuksiin, joista on nyt muotoiltu asiakaskohtainen ratkaisu, tietenkin asiakkaan tarpeisiin vastaten. Työn tekeminen voidaan aloittaa vasta kun spesifinen ohjeistus (toisin sanoen työnanto) on tarkennettu myynnin perusteella tarpeeksi pieniin osasiin projektin vetäjän puolesta, joista osataan muodostaa kehitettävä terminaalitoteutus tuotantoon. Ajan saatossa kuljettajaterminaalitoteutuksista on muodostunut vakiokonfiguraatio, jota ylläpidetään versionhallinnassa. Tämä sisältää käytännön kokemuksen perusteella kehitetyt ratkaisut, jotka palvelevat kehittäjää asiakaskohtaisen toiminnan kehityksessä. Haasteet kehitystyössä eivät muodostu versiohallinnoidusta mallista tai sen käsittelystä. Käyttöpaikkakohtai-

set konfiguroinnit asiakkaan alati muuttuviin vaatimuksiin projektin edetessä voivat aiheuttaa järjestelmälle uudenlaisen kompleksisuuden tason, jota ei projektin suunnitteluvaiheessa pystytty arvioimaan. Projektikehitys suoritetaan ketterällä mallilla, täten räätälöiden asiakkaan näkemys osaksi tuotettua palvelua (Ketterän kehityksen opas 2020).

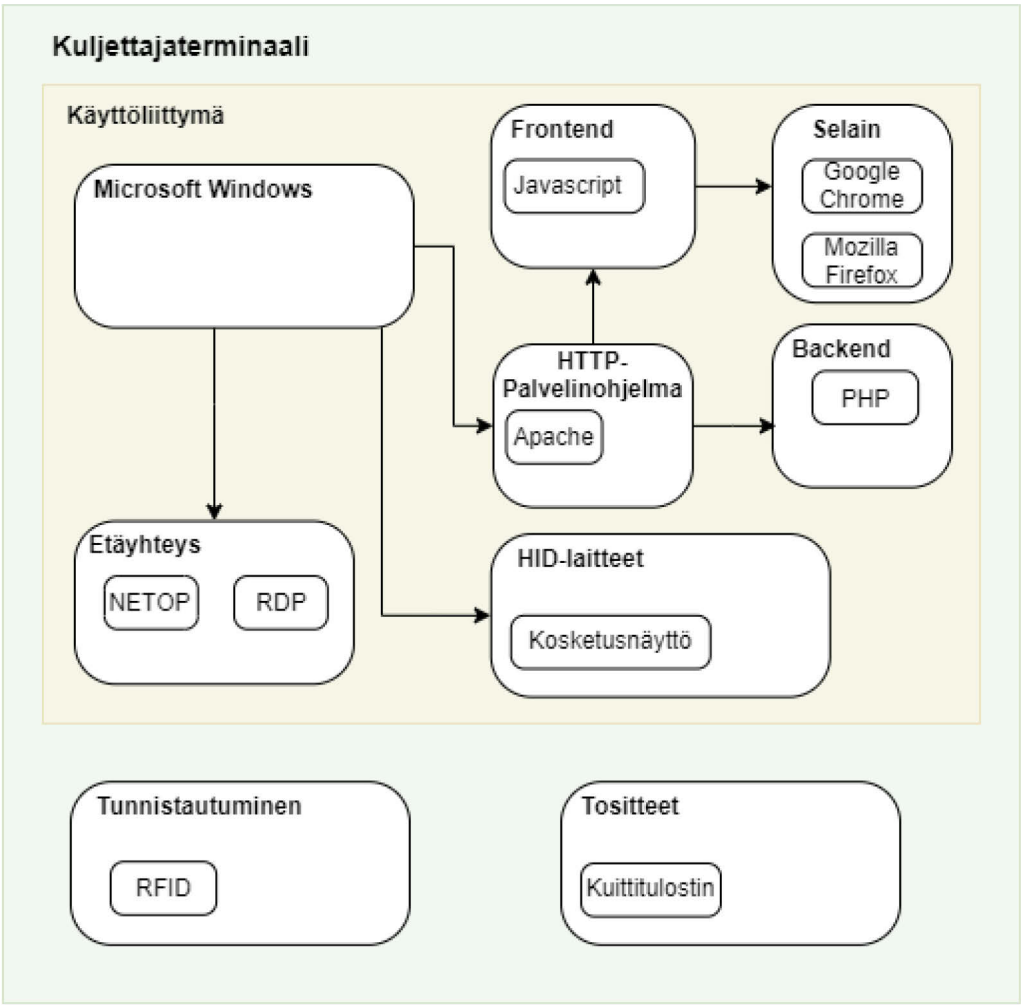
Nykyiset laitteistoläheisen ohjelmiston parissa toimivat kehittäjät Once by Pinja -tuoteperheessä (Once by Pinja, Local and Mobile -ohjelmistokehittäjät, 10/2020) ovat listanneet nykyisen järjestelmäkonfiguraation haasteita kuljettajaterminaalin ympärillä (taulukko 1).

TAULUKKO 1. Nykyisen toteutuksen haasteet

Haaste	**Miksi**
Nykyisen terminaalin vakiokonfiguraatio käyttää Microsoft Windows käyttöjärjestelmää.	Käyttöjärjestelmän konfiguroitavuus ja hallinta hieman haastavaa. Ulkopuolisen hallintaohjelmiston tarve. Osoittanut epästabiilisuutta ympärivuorokautisessa käytössä.
Nykyisen terminaalin käyttöliittymän toiminnallinen konfigurointi.	Koodikonventiot vanhanaikaisia sekä ohjelmisto ylläpidettävyydeltään heikkoa.
Nykyisen toteutuksen virheenkäsittely.	Virheiden selvittäminen on erittäin suuri osa palvelutyötä, tarvitsisi paremman selityksen virheen lähteelle. Liian monta eri tasoa virheille.
Nykyinen toteutus hankala konfiguroida.	Nykyinen on kaukana ajatuksesta, jossa ominaisuudet vain toimivat painamalla ne päälle konfiguraatiosta.
Nykyinen rakenne ei käytä ORM tietojäsentelymallia (objekti sidottu tietokantaan).	ORM on todella kätevä, jos on käytössä olio-ohjelmoinnin mukainen datan jäsentely. Nykyinen tietorakenne ei ole tarpeeksi yhdenmukainen eri käyttöpaikkojen välillä.

3.3 Kehityskohteet

Tärkeä osuus tuotteen jatkokehityksessä on tämän paikallisen palvelukerroksen etähallinta. Tuotannon tilat ovat kielletty ulkopuolisilta, sekä kuljettajaterminaalin fyysinen sijoitus on hankala paikalliselle huoltotyölle. Etähallintaan on olemassa monia ratkaisuita. Microsoft Windowsiin sisäänrakennettu Remote Desktop protokolla on mainio Microsoftin tuoteperheen hallintaan, mutta se ei sisällä nykyisen kirjautuneena olevan käyttäjän näkymän etähallintaa: vain ja ainoastaan uuden istunnon aloittamista etäyhteysoikeudella varustetuin käyttäjätunnuksin, joka on tietenkin jo itsessään riittävä paikallispalvelimelle, johon ei ole liitettynä ulkoista terminaalinäkymää. Tämän takia Microsoftin RDP ei sovellu kuljettajaterminaalin näkymän hallintaan ja on täten korvattava jollain vaihtoehdolla, joka täyttäisi vaatimukset, joihin lukeutuu nykyisen kirjautuneen käyttäjän näkymän etätiedustelu sekä tarpeen mukainen hallinta. Kolmannen osapuolen sovellukseen nojautuminen talon sisäisen ratkaisun puuttuessa on nykyinen malli.



KUVA 5. Palvelukerros kuljettajaterminaalilla

Kehityskohteisiin lukeutuvat kaikki kuljettajaterminaalilla sijaitsevat palvelukerroksen osat. Asiakkaalle rakennettu palvelukerros kuljettajaterminaalilla on aina asiakaskohtaisesti kehitetty, mutta tämä ratkaisumalli ei ole edullinen tuotetta tarjoavalle yritykselle. Yksittäiset osat, jotka ovat kuljettajaterminaalin käytössä, ovat yleisesti ottaen muunneltavissa asiakkaan tarpeitten mukaan, vain jos nämä osat eivät riko yleistä toiminnallisuutta, mikä alun perin on myydyn tuotteen edellytys. Seuraavaksi käydään läpi yksittäin palvelukerroksen osat, mitkä ovat kehityksen tarpeessa kuljettajaterminaalin käyttöliittymätoteutukseen liittyen.

3.3.1 Käyttöjärjestelmä

Käyttöliittymä toimii Microsoft Windows -käyttöjärjestelmässä ajetussa selaimessa. Windows-käyttöjärjestelmän ongelmat ympärivuorokautisessa ajossa tuottavat päänvaivaa, kun tarkoitus on pitää yllä käyttöliittymänäkymää ilman häiriöitä niin visuaalisesti kuin toimintavarmuudellisesti, samanaikaisesti huoltoyhteydenottojen sekä kantayhteyden jatkuvan ylläpidon kanssa. Käyttöjärjestelmä sisältää paljon toiminnalleen kriittisiä palveluita, jotka yhdessä tuottavat raskaan pinon yksinkertaiseen käyttöön tarkoitettuun paneelitietokoneeseen. Käyttöjärjestelmän hyviin ominaisuuksiin kuuluu kolmannen osapuolen tuki, mutta tämä aiheuttaa lisätöitä ylläpitää käyttölisenssejä ylläpitoon sekä kehitykseen vaadittavista ohjelmistoista. Windowsin suurin heikkous ympärivuorokautisessa käytössä on käyttöjärjestelmän taipumattomuus palveluiden päivityksiin ilman koko järjestelmän alasajoa, sekä lukuisat ajonaikaiset järjestelmän kaatavat virheet (Top 10 Windows Anomalies n.d.). Kuljettajaterminaali on päivityksien aikana käyttökelvoton tuotannolle, mikä johtaa pahimmillaan koko tuotannon seisaukseen.

3.3.2 Frontend

Kuljettajaterminaalin ulkoinen näkymä, graafinen käyttöliittymä, on rakennettu dynaamisesti muodostetuista valmiiksi kehitetyistä komponenteista. Ohjelmointikielenä on käytetty maailman käytetyintä asiakaspuolen ohjelmointikieltä, JavaScriptiä (Historical trends in the usage statistics of client-side programming languages for websites 2020). Tämä näkymä on muotoiltu Cascading Style

Sheet (CSS, rakenteellisten dokumenttien tyyliohje) -kielellä. Tämän pinon konfigurointiin on tehty välikappale, eräänlainen helpottava kerros (Terminaali-UI), joka ottaa sisäänsä tiivistettyjä argumenttijonoja, joissa kerrotaan mikä elementti halutaan saada luotua näkymään sekä mitä toiminnallisuutta tämä elementti sisältää. Tämä näkymän konfigurointi on todella yksinkertaistettua mutta raskasta, koska elementtien määrittely on näkymäkohtaisesti luotava kuljettajanäkymän tietokannassa asti. Tämä tietokannassa tehty määrittely on erittäin hidas prosessi näkymän luomiseksi, kun se sisältää elementin määrittelyn, joka viedään uudeksi tietokantariviksi, jonka testaus toimii vain uudelleen ladattaessa tämä näkymä internet-selaimessa. Itse selainriippuvuus on myös tämän pinon yksi heikkous. Tuotteen historian aikana on toteutettu käytännön testejä eri selaimilla, mutta mikään kolmannen osapuolen toteutus internet-selaimesta ei ole ollut ongelmaton.

3.3.3 Backend

Taustalla tietokantaan yhteydenmuodostajana toimii PHP ohjelmointikieli, jossa ohjelmakoodia tulkitaan vasta ohjelman suoritusvaiheessa. Yhteyden muodostus tietokantaan tiedon tarjoilemiseksi graafiselle käyttöliittymälle konfiguroidaan tähän taustapalveluun. Ongelmaksi nousee virheenkäsittely tuotannossa, kun taustapalvelun kantayhteys epäonnistuu. Graafisen käyttöliittymän elementit rakennetaan tietokannan ohjeiden mukaisesti, mutta kun tietokantayhteys on saavuttamattomissa, on sovelluksen ajo täten mahdotonta.

Web-palvelimen roolissa Apache tuottaa näkymän sekä mahdollisia rajapintoja tarpeellisille lisäyhteyksille kuljettajaterminaaliin. Apachen ylimääräinen konfigurointi aiheuttaa eriäviä toteutuksia tuotantoon, sekä sen ylläpito PHP:n rinnalla tuottaa ylimääräistä rasitetta, kun yleinen hyöty jää erittäin marginaaliseksi.

3.3.4 Etäyhteydet

Etäyhteydet ovat tuotteen tuen kannalta elintärkeitä. Ilman etäyhteyksiä ei kuljettajaterminaalin tilasta voida varmistua. Fyysinen pääsy kuljettajaterminaaliin on estetty normaalissa käyttötilanteessa. Asiakkaalle annetaan pääsyoikeudet

terminaaliin sen huoltoon liittyvissä toimenpiteissä, mutta tämä on aina tuotannon pysäyttävä toiminta. Kuljettajaterminaalin tilan tiedusteleminen asiakkaalta on asiakasta kuormittavaa ylimääräistä työtä.

Etäyhteyksissä kuljettajaterminaaliin tarvitaan nykyinen näkymä asiakkaalle, kohdetietokoneen korkean tason hallinta, sekä tiedostojen siirto internetin ylitse. Käytössä etäyhteyksiä varten on käytössä Netop:n tuottama etäkäyttöön tarkoitettu Netop Remote Control -sovelluspari. Kohdekoneessa etäyhteydelle pidetään yllä palvelua (Host), jonka tarkoituksena on tarjota etäkäyttöön paikallisille työkaluille rajapinta (Guest). Tämä on erittäin kattava tuote etäyhteyksille ilman häiriötä kohdetietokoneessa, mutta niin kuin Microsoft Windowsinkin osalta, lisensointi on kallista sekä ongelmatilanteet ovat erittäin haastavia ratkaista. Etäyhteyksissä ongelmia tuottavat tiukat suojauskäytänteet yhdistettynä käytettyihin ohjelmiin kolmannelta osapuolelta.

4 KULJETTAJATERMINAALI TULEVAISUUDESSA

Ohjelmiston kehitys ei saa aiheuttaa asiakkaalle lisäkustannuksia pidemmälläkään aikavälillä. Ohjelmiston ylläpito sekä päivitykset kuuluvat tuotteen maksullisiin ominaisuuksiin, joten kaikki ylimääräinen virhetilanteiden ratkaiseminen on pahimmassa tapauksessa itse ohjelmistokehityksen hidaste (kun 1. tasolta, helpdeskistä, edistetään ongelma 2. tasolle, kehittäjäpuolelle). Kehittäjän työhön sisältyvät tuotekehityksen rinnalla uusien tai olemassa olevien asiakasympäristöjen kehittäminen sekä erikseen maksullinen ylläpito. Tarkastellaan seuraavaksi mitä ohjelmistotuotanto palvelee nykyään, sekä miten sen pystyisi valjastamaan tämän tuotteen hyödyksi tulevissa kuljettajaterminaali toteutuksissa.

4.1 Ohjelmistokehityksen nykytila

4.1.1 Moderni ohjelmistokehitys

Informaatioteknologia pohjautuu vahvasti tiedonvälitykseen, jossa nykyään arvostetuimpiin ominaisuuksiin kuuluu luotettavuus, kuluttajan turvallisuuden tunne, että tiedon kulku on vain tiedettyjen päätelaitteiden välistä, eikä sitä pysty selkokielellä inhimillisillä tavoilla tulkitsemaan.

Modernissa ohjelmistokehityksessä tiedon ajantasaisuus on erittäin kriittinen kulmakivi, mitä ohjelmiston luotettavuuteen tulee. Modernissa ympäristössä ei tapahdu (tai oletettavasti ei pitäisi tapahtua) keskustelua niinkään ohjelmiston kehitysympäristöistä tai käytetyistä toteutusteknologioista, mutta sen sijaan ohjelmointikäytänteistä sekä hyvästä ohjelmointietiikasta. Nämä käytänteet, jotka todetaan useamman esimerkin valossa hyviksi, pyritään dokumentoimaan tavalla, jolla pystytään samanaikaisesti todentamaan, että säilyttämään tämä tieto sitä tarvitseville. Tämän voi huomata alan alati päivittyvästä tietokirjallisuudesta, kun ensimmäinen painos ei tiedettävästi ole koskaan saavuttanut ohjelmointioppaissa viimeisimmän painoksen leimaa. Myös uusien ja aiemmin tuntemattomien käyttötapausten myötä ohjelmointia opettavien tai taltioivien tahojen on lisättävä näiden uusien tapauksien tuomista kokemuksista näkemys portfolioihinsa. Kaiken kaikkiaan onnistunut ohjelmistoprojekti on lähestulkoon aina kaksiteräinen miekka, samalla kun projektin haluttu tulos on käytännössä valmis, eli

tuotos palvelee toivotulla tavalla ilman tuotannosta aiheutuneita virheitä, niin myös epäilemättä tuotoksessa on jokin osa-alue, joka on jo todettu kehnommaksi menetelmäksi kuin jokin toinen tai mahdollisesti käytetty kirjasto sisältäisi vasta havaitun tietoturva-aukon, jonka haitallinen hyödyntäminen voi olla ollut jo käytännössä pidempäänkin valloillaan (Hacking the PS4. 2015).

Eettisyys ohjelmointialalla pyritään tiivistämään jokaisen yrityksen omaan Code Of Conduct – eettiseen ohjeistoon, itsenäiseen dokumenttiin, joka ohjaa organisaation toimintaa kohti asetettuja tavoitteita. Tämä mahdollistaa muun muassa lojaaleja sekä kestäviä työntekijä- ja asiakassuhteita. Asiakkaiden ei missään tilanteessa täytyisi joutua epäilemään yrityksen kehittämän järjestelmän eettisyyden tai luotettavuuden tasoa, mitä tulee sopimuksen aikaisen selvityksen myötä myytyyn kokonaisuuteen. Asiakas tekee sitovan päätöksen tuotteesta ostohetkellä, mutta nykyohjelmistokehityksessä järjestelmää tuottavan yrityksen täytyy olla tietoinen kehitettävään järjestelmään liittyen vallitsevasta tilasta, varsinkin jos tuote hyödyntää vallitsevia teknologioita, joista hyvänä esimerkkinä kommunikaatioteknologiassa vaaditaan tarkkuutta sekä eheyttä tiedonsiirrossa tietoturvaan ja tietosuojaan liittyvien uhkien takia.

4.1.2 Moderni ohjelmiston käyttökokemus

Käyttökokemus (User Experience, UX) kulkee käyttöliittymän (User Interface, UI) rinnalla, luoden kokonaisuuden, jota voidaan mitata käytettävyydellä (halutun lopputuloksen saavuttamisen helppoutta) sekä millä tuotetta ylipäätään voidaan kokea. Näistä kahdesta kuitenkin keskitytään työssä enemmän käyttökokemuksen valjastamiseen kuin itse ulkonäköseikkoihin, jotka yleensä kulkevat valtavirran mukana. Käyttäjän saama kokemus ohjelmiston käytöstä on todennäköisesti keskeisin tekijä siihen, kuinka asiakkaalle syntyy halu sitoutua tuotteen käyttöön. On myös tärkeä saada käyttäjä perehdytettyä tuotteen käyttöön johdonmukaisen käyttöliittymäsuunnittelun sekä tuotteen tarkoituksenmukaisen toimintamallin avulla. Tähän nojaten ohjelmistoja ei pitäisi lähtökohtaisesti suunnitella teknologian ehdoilla vaan käyttäjien. On täten erittäin tärkeää tuntea asiakaskunta, jolle tuote on suunnattu. Käyttökokemuksen kehittäminen vaatikin aika-ajoin kommunikointia itse tuotteen kuluttajan kanssa tuotannossa, jotta

käyttökokemuksen kehittämisen suuntaa voidaan arvioida analyyttisesti. Ohjelmistoja rakennetaan huviksi sekä hyödyksi, taiteeksi sekä tieteeksi. Ohjelmiston lokalisaatio ja käyttäjien kulttuuriset erot johtavat mahdollisesti hämmennykseen, jos vaatimuksissa ei ole otettu huomioon tarpeeksi laaja-alaisesti käyttäjäkunnan diversiteettiä. Tämän takia on hyvin hankala päätellä, miten voidaan luoda kaikille käyttäjille yhdenvertainen onnistunut käyttökokemus. Tätä tukemaan on luotu käyttökokemuksen laadulle ehtoja luetteleva standardi (SFS ISO 9241, 13), joka sisältää vaatimukset näyttöpäätteellä tehtävästä toimistotyön ergonomiasta.

4.1.3 Tiedon käsittely

Data muodostuu informaatioksi, kun sitä käsitellään tietoa muodostavassa ympäristössä. Tämä mahdollistaa sensitiivisen tiedon luomisen, joka on ohjelmistoalan yksi suurimmista haasteista. Oletetusti järjestelmä kehitetään käsittelemään tietoa sillä tavoin, että ohjelmisto palvelee käyttäjää työkaluna, käyttäjä antaa tietoa, jonka järjestelmä prosessoi tavalla, jolla se tuottaa käyttäjälle halutun lopputuloksen. Elektroninen laskin on yksinkertaistettu esimerkki tästä. Käyttäjä haluaa tietää kahden arvon välisen suhteen valitun operaattorin mukaisesti. Käyttäjän täytyy omalla toiminnallaan syöttää laskutoimituksen eri osat laskimelle, jotta laskin voi tulostaa käyttäjälle tuloksen. Tässä tapahtumasarjassa käyttäjä on antanut laskimen käyttöön itse syöttämänsä tietoa, joka ei ole yleisesti saatavilla missään, mutta on nyt käyttäjän ja laskimen välisessä kommunikaatiossa käytettyä. Yksinkertainen laskin ei sammuttuaan kykene muistamaan käyttäjän viimeksi syöttämiä tietoja, mutta kehittyneempi laskin voi sisältää lisäominaisuuden, jotta käyttäjän toiveesta pitkäaikaiseen muistiin tallentuu käyttäjän itse määrittelemää tietoa. Pitkäaikaismuistiin tallennettua tietoa voidaan hyödyntää useissa eri tilanteissa ja se on täten lisäarvoa tuottava elementti, koska täten tiedon syöttö on tapahtunut vain kerran, eikä laskimen käyttäjän ole mahdollisesti enää tarve edes muistaa tätä tietoa, kun se on pikanäppäimestä helposti saatavilla.
Ohjelmistotuotannossa data oletetaan tulevan ulkoisesta lähteestä. Käyttäjälle rakennetaan mahdollisuus antaa syötettä hänelle tutummassa muodossa, josta tämä tieto otetaan käsittelyyn lopputuloksen laskennassa. Tämä tieto täytyy säi-

löä ajonaikaisesti käytettävään muistiin, johon voidaan viitata ajon aikana. Nykyään tämä tieto voi olla aivan mitä vain, sekä tallentua aivan maailman toiselle puolen. Käyttäjälle on annettu lähes rajattomat vaihtoehdot syöttää henkilökohtaisia tietoja järjestelmään, sekä järjestelmät kehottavat useammassa tilanteessa myös antamaan todenmukaiset tiedot erilaisten palvelujen järjestämiseksi. Tiedon käsittelyssä on täten huomioon otettava vallitsevat tietosuojalainsäädännöt sekä inhimilliset rajat henkilökohtaisten tietojen käsittelyyn ajon aikana sekä tallennuksessa. On tyypillistä, että kaikki data ei ole samanarvoista. Hyvien käytänteiden mukaista on salata, sekä rajoittaa pääsyä tähän dataan sille kuulumattomilta tahoilta. Datan käsittely ohjelmiston ajonaikaisesti ei saisi tuottaa käyttäjälle tunnetta, että hänen syöttämänsä tieto olisi annettu järjestelmälle hänen puolestaan väärin.

Tiedon käsittelyyn kuljettajaterminaalilla kuuluu etukäteistiedon saaminen kuormasta, ja tämä on tärkeä asia silloin, kun kuljettaja tulee antamaan vähemmän merkityksellistä tietoa, joka vähentää virheen mahdollisuutta. Kuljetustehtävät ovat kuitenkin suurimmalta osin kuljettajille määrättyjä valmiita kokonaisuuksia, jossa kuljettajan tehtävä on vain kuljettaa materiaali vaadittuun määränpäähän. Tämän tehtävän päätyminen mahdollisimman kattavasti vastaanottopaikkaan minimoisi kuljettajalta vaadittuja tietoja kirjausvaiheessa täten vähentäen virheitä tuntuvasti.

4.2 Palvelun päämäärä

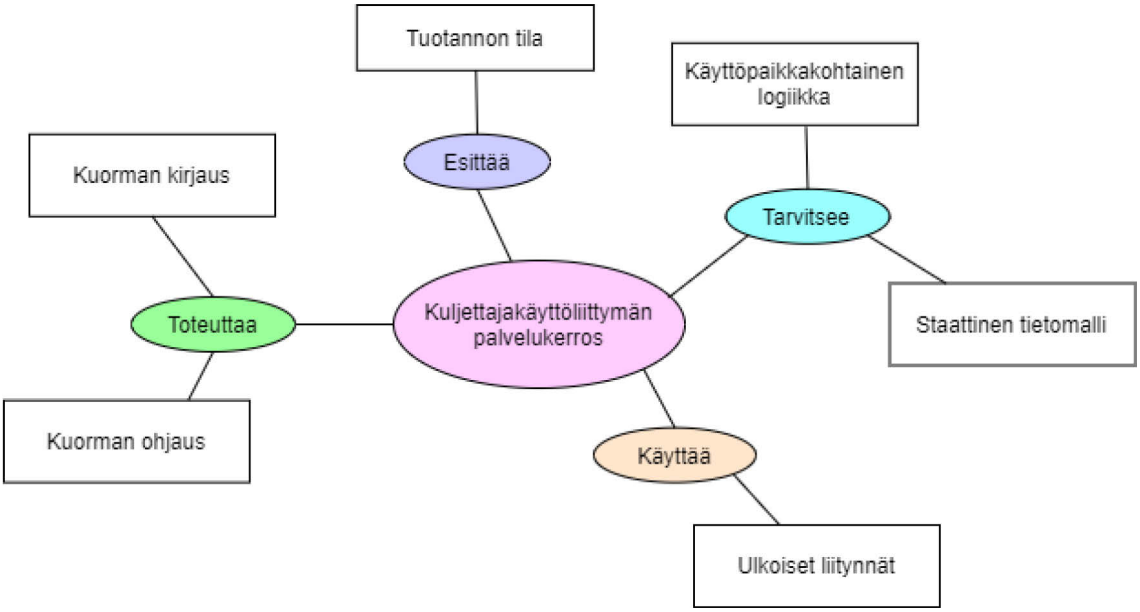
Kuljettajaterminaalin tarkoitus on tehdä tuotannon tietojenkeruu energiakuormittain antaen ylemmälle tasolle kuvan siitä, minkälainen materiaalivirta kulkee käyttöpaikkakohtaisesti. Tuotannossa voidaan määritellä ennalta saapuvan kuorman ehtoja, jotka ovat linkitettynä mahdollisesti kuljettajan tapaan tunnistautua kentällä. Jokainen kirjattu kuorma pohjautuu tuotannossa tehtyyn sopimukseen, jossa määritellään mahdolliset aineet, liikennöitsijät, toimittajat sekä ajoneuvot, joilla sopimuksen mukaiset energiakuljetukset järjestetään. Kuljettajan tärkein työ on palvelun oikeellisuuden takaamiseksi syöttää järjestelmään tiedot kuljetettavasta kuormasta, käyttöpaikkakohtaisesti mahdollistetun tiedon-

syöttötavan avulla. Tämän työn tarkoituksena on kartoittaa vaatimukset uudenlaisille tavoille syöttää tietoa käyttämällä nykyistä järjestelmää pohjatoteutuksena.

KUVA 6. Paikallisen järjestelmän palvelumalli korkean tason kuvauksena

4.3 Tarvittavien palvelukerrosten määrittäminen tuotannossa

Kuljettajaterminaalin täytyy sisältää mahdollisuuden tuotannossa: esittää käyttöpaikkaan liitettyjä tietoja reaaliaikaisesti, tuotannon tarpeiden mukaisesti (Ulkoisten liityntöjen esittäminen). Lisätä sekä muokata dataa rinnakkaisten tietovarastojen välillä (Kuormien sekä ulkoisten liityntöjen hallinta). Tarjota energiakuljetusten kuljettajille selkeä näkymä, jonka avulla voi kirjata kuorman käyttöpaikalle tai käyttöpaikalta sekä varmistua tapahtumasta tosittein. Reaaliaikaisen hallintanäkymän tai -linjan huoltotilanteita varten, josta ei aiheudu keskeytyksiä tuotantoon, sekä tarjota ympärivuorokautista palveluvarmuutta



KUVA 7. Kuljettajakäyttöliittymän vaatimushahmotelma

4.3.1 Tietojen esitys

Esitysteknisesti terminaali on ajansaatossa ollut aina lähes samankaltainen fyysisiltä ominaisuuksilta. Kosketusnäyttöpaneelilla varustettu tietokone, jonka monitorille mahtuu samanaikaisesti lähietäisyydeltä luettava teksti, sekä ruudulla esitettävät, inhimillisesti tuotannon olosuhteisiin nähden ihmiskäden kosketuksen tunnistavat painikkeet. Tietokoneeseen liitetyn näytön tarkkuuden sekä koon merkitys on riippuvainen kerralla esitettävän tiedon määrästä. Nykyinen toteutus antaa mahdollisuuden käyttäjälle vierittää näkymää esitettävän tiedon ylittäessä esitystekniset rajat. Ottamalla nämä nykyiset rajoitteet huomioon suunniteltaessa tulevaisuuden toteutuksia, on mahdollista kehittää kustannustehokkaita esitysteknisiä ratkaisuja, joissa keskitytään tiedon suppeaan tarjontaan vaatimuksien mukaista toimintaperiaatetta noudattaen (kuva 7), joka antaa mahdollisuuden kuormien kirjaukseen häiriöttä.

Kuorman kirjauksen sekvenssi mallinnetaan tietojen esitykseen valitussa muodossa kuljettajaterminaalin näytöllä ajonaikaisesti, ja mikä on tärkeä elementti kehityksessä. Koska nykyinen malli näkymien konfiguroinnissa ei ole sidottu yhteen rajapintaan, on täten jätetty liian suuri virheen mahdollisuus näkymien konfiguroinnissa kehittäjän harteille toteutuksen laajuuden muuttuessa.

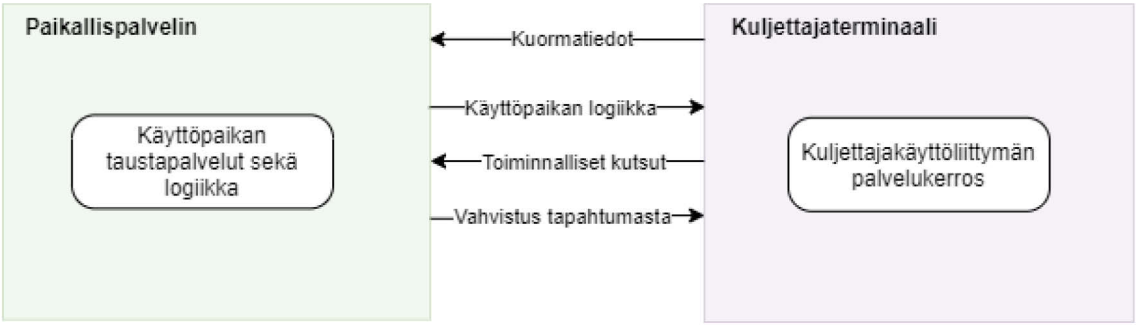
4.3.2 Datan keräys

Kuljettajaterminaalilta saatava data on joko linkityksien kautta jäsenneltyä tietoa, joka muodostuu valitsimiksi käyttäjälle, tai käyttäjän itse antamaa tietoa käyttöliittymästä käsin syötettynä. Tämän tuloksena syntyy kuorman luonti -tapahtuma, jossa tieto tallennetaan vakiomuotoiseksi kuormariviksi, josta järjestelmässä liipaistaan tieto, että käyttöpaikalla on luotu energiavirtaan vaikuttava tapahtuma. Tämä tapahtuma on ajoneuvokohtainen ja järjestelmä täten vaatii, että ajoneuvo toteuttaa sekvenssin sekä tulo- että lähtöpuolen kirjauksesta, joka tapahtuu tälle samaiselle kuormariville päivityksenä. Jotta datankeruu kuljettajaterminaalilla onnistuisi, on varmistuttava, että tieto mitä käyttäjältä tullaan vaatimaan, on energiakuljetusta järjestävän tahon tiedossa tai muuten saatavilla laitoksen käytön osalta. Muut ulkoiset linkitykset ja logiikka paikallispalvelimelta (mukaan lukien käyttöpaikkakohtainen tunnistautuminen) käsiteltävään kuormaan tuovat oman

käsittelykerroksen, joka täytyy ottaa huomioon käyttöpaikkakohtaisessa sekvenssissä, eikä täten vaadi enää kuljettajalta huomiota itse kuljettajaterminaalitoteutukseen.

4.3.3 Ohjelmiston tarjonnan määrittäminen

Tuotteen ohjelmistokehityksessä huomion täytyisi aina pysyä ennestään kehitetyn ohjelmiston lisäarvoa tuottavassa toiminnassa (pakollisten ylläpidollisten tarpeiden lisäksi). Mitä tuotteen myytäviin ominaisuuksiin tulee, on vastoin hyvää käytäntöä myydä ominaisuutta, jota ei ole vielä kehitetty (tai tuotteistettu tarpeellisilla edellytyksillä). Täytyisi aina pyrkiä tuotteistamaan myytävä ominaisuus jo etukäteen, esim. lähteä mielikuvalla myymään toteutusta, josta keskustelu jatkuisi kehitystiimin sekä projektihallinnan sisäisiin iteraatioihin ongelman ratkaisemiseksi ominaisuutena. Ohjelmistokehitykselle olisi tarpeellista määritellä strategia, jolla pyrittäisiin aina tarjoamaan asiakkaalle lisäarvoa tuottava ominaisuus ilman riskiä siitä, että asiakas joutuu uudelleen arvioimaan kehitetyn ominaisuuden soveltuvuutta, mikä johtaa usein tilanteisiin, joissa ominaisuutta joudutaan korjaamaan.



KUVA 8. Kuormatietojen syötöstä tapahtuman vahvistukseen kuljettajaterminaalilla

Ohjelmistokehityksessä on otettava huomioon, että lopputuloksena saadaan aikaiseksi luotettava tapahtuma, joka antaa ilmi tilansa vahvistamalla käyttäjän antaman tiedon järjestelmään kirjatuksi (kuva 8).

4.3.4 Ylläpidolliset tarpeet

Kehityksessä usein dokumentaation taso jää mukautuvissa ominaisuuksissa vähemmälle työn mahdollisesti monimutkaistuessa ja pitkittyessä. Ylläpidon kannalta kriittisin tarve on päästä mahdollisimman pienellä kitkalla tuotannon konfiguraatiosta selville, jotta ylläpidolliset toimenpiteet pystytään järjestämään ilman ylimääräisiä kustannuksia aiheuttavia tekijöitä. Ylläpidon tarpeet kuljettajaterminaalilta kulminoituvat tietoliikenneyhteyksien toimivuuteen. Tällä hetkellä tuotteen paikallisen osuuden suurimmat haasteet ovat fyysiset laiteviat, jotka aiheuttavat mahdollisia aukkoja ongelmanratkonnan tasolla. Ylläpidollisesti tärkeintä huoltotilanteita varten tulisi olla reaaliaikainen näkymä tuotannon tilan tarkkailua sekä hallintaa varten, jotta voitaisiin varmistua terminaalin toiminnasta ilman ylimääräistä keskeytystä tai muuta aistittavaa häiriötä tuotantoon.

4.3.5 Terminaalitoteutuksen ohjelmiston jakelu

Terminaalipäätteen jakelu on jaettu kahteen osaan, fyysiseen toimitukseen sekä ohjelmistoasennuksiin. Keskitymme nyt vain ohjelmiston jakeluun, fyysisen puolen toteuttaessa pääosin ulkoiselta olemukseltaan sekä laitevalinnoillaan tuotannon vaativien olosuhteiden palvelemista, mukaan lukien kolmansien osapuolien integroinneista johtuvien seikkojen takia.

Terminaalipäätteen ohjelmiston jakelu on toteutettu valmiskokoonpanolla, joka sisältää kaiken tarpeellisen vakiokonfiguraation mukaisesti, ja jota päivitetään tarpeen tullen mahdollisien kehitystarpeiden sitä vaatiessa. Ohjelmisto on valmiiksi asennettu sekä konfiguroitu tuotannonmukaiseen tilaan ennen käyttöpaikalle asentamista. Kuljettajaterminaalin ohjelmiston osalta ei ole järjestetty automatisoituja ratkaisuita, jotka helpottaisivat omalta osaltaan tuotteen tämänhetkisen tuotannon ohjelmiston tilan katselmointia sekä kehityksessä tehtävien julkaisuiden käyttöönottoon liittyviä toimia. Nykyinen malli on lähtökohtaisesti ajateltu kerran toimitettavaksi kokonaisuudeksi, jonka jälkeen päivitykset tapahtuvat osana suunniteltuja huoltokatkoja tuotantoon. Käyttökatkokset tuotannossa ovat ainoa tapa päivittää kuljettajaterminaalin ohjelmistoa ilman, että ohjelmistopäivityksestä aiheutuva käyttökatkos terminaalilla aiheuttaisi tuotantoon merkittävää haittaa.

5 POHDINTA

5.1 Käyttökokemuksen takaaminen

Tuotannossa ympärivuorokautinen mahdollisuus kuormien kirjaukselle järjestelmään on kokonaisuus, joka vaatii järjestelmältä luotettavuutta, joka on järjestetty vaatimusten mukaisen kehityksen avulla. Lukuun ottamatta fyysisiä laiterikkoja tai verkkoyhteyden saavuttamattomuutta järjestelmän vastinpäähän, terminaalipäätteen on tarjottava saumatonta tiedon esitystä sekä sulavaa että selkokielistä käyttöliittymää käyttäjälle. Käyttöliittymän on esitettävä kuljetuksen kirjaukseen liittyvät oleelliset tiedot sekä kuljettajan valintojen mukaiset mahdolliset käyttöpaikkakohtaiset toiminnot. Mahdollisen virheen sattuessa kirjaustietojen syötössä, käyttäjälle täytyy antaa mahdollisuus peruuttaa tai valita uudelleen syötettävä tieto. Jos virhetilanteen aiheuttaja on mahdollinen normaalia tietojensyöttötoimintaa estävä tapahtuma, täytyisi järjestelmän pystyä palauttamaan tila ennen aiheutunutta virhettä. Jos taas virhetilanne johtuu ulkoisista tekijöistä tai käyttöpaikkaan sidotun logiikan konfliktista, on täten annettava käyttäjälle selvästi esitettynä syy sekä jatkotoimenpiteet ongelman itsenäiselle ratkaisulle tuotannossa, ellei tapahtuman korjaus ole mahdotonta tuotannossa kuljettajaterminaalin käyttäjälle (ohjelmiston palautumaton virhetila). Tulevaisuudessa käyttökokemus nähtäisiin enemmän sulavana toimintana käyttöpaikalla, jonka virheiden korjaukset mahdollisuuksien mukaan olisi järjestettävissä lokitietoihin perustuvasta tapahtuman oikaisusta, tapahtumaan liitetyllä varmennetulla, juoksevana tallennetulla tiedolla.

5.2 Tiedon käsittelyn tarve

Kuormien kirjaus käyttöpaikalle sekä käyttöpaikalta on kuljettajaterminaalin tärkein tehtävä. Kirjattavan tiedon täytyy perustua todellisiin tapahtumiin tuotannossa, jotta energiavirtojen raportointi pysyisi eheänä koko kiertokulun aikana. Tietojen kerääminen tuotannosta käyttöpaikalla tapahtuvan toiminnan takia on suotavaa, jotta asiakkaan on mahdollista todentaa tapahtumat kentällä ja täten hyödyntää kerättyä tietoa mahdollisesti tuotannossa tapahtuvan toiminnan kehittämisessä. Tarve tiedon käsittelyyn sekä varastointiin on täten kaksiosainen. On

vaatimus energiavirtojen raportointiin, että toiminnan ylläpitämiseen, sekä vaatimus pystyä todentamaan tapahtumat ja sekvenssin tapahtumien oikeellisuus. Nämä kaksi, kun voidaan varmistaa aina saataville, on mahdollista täyttää tiedon keräämiseen liittyvät tarpeet.

5.3 Tapahtuman hallintaketju

Ongelmatapauksista tuotannossa vastaa osin myös ulkoiset liitynnät paikallislogiikkaan, jotka sisältävät yhtä lailla virheen mahdollisuudet kirjaustoiminnassa. Huomioon otettavaa onkin varmuus tuotannon tilasta sekä selkokielinen vastaus vajaatoiminnan selvityskyselyyn. Nykyisessä järjestelmässä ollaan vahvasti riippuvaisia lokitietojen antamasta viimeisimmästä järjestelmän ulkoisen liitynnän ohjauksen tilasta, joka sekin täysin tai ainakin osittain on kehittäjän vastuulla, implementoidaanko juuri tämän kyseisen muuttujan tilatietoa historiatietoihin. Yhä enemmän todetaan tarvetta automatisoidulle tapahtumien hallintaketjun organisoimiselle, joka muodostuisi työkaluksi tilatietojen monitorointiin ja hallintaan.

5.4 Ohjelmapinon keskittäminen sekä jakelu

Ohjelmistopino kuljettajaterminaalilla on mahdollista kehittää vaatimuksen mukaiseksi yksittäiseksi kokonaisuudeksi, joka tarjoaa nykytekniikoilla toteutettuna mahdollisuuden, sekä selkeästi suoraviivaisempaan ohjelmistokehitykseen että yksinkertaisempaan ohjelmiston jakeluun. Tuotannon toteutuksiin on järjestettävä selkeämpi kokonaisuus ylläpidollisesti, joka on mahdollista uudenlaisen määrittelyn avulla. Kuljettajaterminaalilla sijaitsevalle ohjelmistolle täytyisi asettaa selkeämpi suuntaviiva, mikä pelastaisi terminaalin konfiguroinnin suuresta osaa sen nykyisistä sudenkuopista. Terminaalitoteutuksen pohjaratkaisun vahvuudet ovat ajansaatossa muuttuneet tuotetta hankaloittaviksi tekijöiksi.

5.5 Päätelmä työn kattavuudesta

Työn sisältö kattaa korkean tason kuvauksen sekä analysoinnin kuljettajaterminaalilla sijaitsevasta ohjelmistopinosta. Työ on pyrkinyt vastaamaan siihen, mitä pinon täytyisi pystyä tuotannossa toteuttamaan, mikä toimisi vähintäänkin pohja-

tietona uudenlaisen arkkitehtuurin kehittämiseen. Työ ei vastannut tutkimusosuuteen, jossa olisi lähdetty suunnittelemaan vaihtoehtoisien mallien mukaisesti nykyistä toteutusta, vaan jää sen osalta toimeksiantajalle itselleen. Työn arvo toimeksiantajalle kehittyy haluttuun tasoon viimeistään siinä pisteessä, kun työn sisältöä hyödynnetään uuden toteutuksen kehitystiimin suuntaviivana sekä perehdyttämisessä.

Työn rajauksissa otettiin huomioon työn toimeksiantajan halu saada yhteenveto terminaalipäätteen vaihtoehtoisen toteutuksien mahdollisuuksista esim. uudenlaisella arkkitehtuurikuvauksella, joka ei kuitenkaan päätynyt työn sisältöön asti. Toimeksiantajalle pitäisi työstä silti syntyä hahmotelma, mitä tuotannon tarpeita todellisuudessa kuljettajaterminaalin näyttöpäätteellä sijaitsevasta terminaalisovelluksesta täytyisi saavuttaa, sekä mitkä ovat suurimmat haasteet nykyisen, toimivan, mutta haastavan mallin tilalle.

LÄHTEET

JHS 152 Prosessien kuvaaminen. Liite 1 Tyypillinen kuorman kirjaus prosessikaaviona. Luettu 20.11.2020. http://docs.jhs-suositukset.fi/jhs-suositukset/JHS152/JHS152.html

Ketterän kehityksen opas 2020. Luettu 24.11.2020. https://www.scrumguides.org/scrum-guide.html

Top 10 Windows Anomalies, Maintenance Headache of Windows. Open Source Software vs. Commercial Software: Migration from Windows to Linux. Luettu 10.11.2020. http://members.apex-internet.com/sa/windowslinux/03-05-reliability-stability.html

Historical trends in the usage statistics of client-side programming languages for websites. W3Techs. Luettu 22.10.2020. https://w3techs.com/technologies/history_overview/client_side_language/all

SFS 9241. 1998. Näyttöpäätteillä tehtävän toimistotyön ergonomiset vaatimukset. Osa 13: käyttäjäopastus. Helsinki: Suomen Standardisoimisliitto SFS.

Hacking the PS4. Introduction to PS4's security, and userland ROP. Luettu 12.10.2020. http://cturt.github.io/ps4.html

LIITTEET

Liite 1. Tyypillinen kuorman kirjaus prosessikaaviona