Tampereen yliopisto

Matti Kinnunen

# PIENYRITYKSEN TYÖNHALLINTAJÄRJESTELMÄ: KEHITTÄMINEN JA ARVIOINTI

Tekniikan ja luonnontieteiden tiedekunta
Diplomityö
Toukokuu 2021

# TIIVISTELMÄ

Matti Kinnunen: Pienyrityksen työnhallintajärjestelmä: Kehittäminen ja arviointi
Diplomityö
Tampereen yliopisto
Johtamisen ja tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Toukokuu 2021

---

Toiminnanohjausjärjestelmät (ERP, *Enterprise Resource Planning*) ovat yritysten tietojärjestelmiä, joilla organisaatiot voivat integroida erilaiset prosessit ja datan käsittelyn yhdeksi paketiksi. Myös työnhallintajärjestelmät luokitellaan toiminnanohjausjärjestelmiksi, mutta niiden käyttötarkoitus ja laajuus tulkittiin tässä työssä täysimittaisia toiminnanohjausjärjestelmiä suppeammaksi. Työn päätavoite on kehittää selaimessa toimiva, toimeksiantajan vaatimuksiin ja suunnitelmiin perustuva työnhallintajärjestelmän perusversio, jota on tarkoitus laajentaa ainakin varastonhallintaan ja laskutukseen liittyvien toimintojen osalta erillisessä projektissa perusversion valmistumisen jälkeen. Toimeksiantaja on jätevesijärjestelmien huoltoihin ja korjauksiin erikoistunut pienyritys. Tavoitteen saavuttamiseksi työssä kehitetään ensin metodologia, jonka avulla kyseinen ohjelmistoprojekti voidaan toteuttaa etänä toimeksiantajalle, jolla ei ole valmiutta tarkastella ja hyväksyä kehitettävän sovelluksen kehitysvaiheita pelkän lähdekoodin avulla. Metodologian kehittämisen kannalta parhaaksi löydetyistä tietolähteistä arvioitiin tutkimus, jossa kehitetään metodologia websovelluksen kehittämiseksi suoraan etäpalvelimelle ilman kehittäjien tietokoneille konfiguroitavia kehittämisympäristöjä. Metodologiassa käytetään verkossa toimivia koodieditoreja palvelinkoodin editointiin ja vastaavia ohjelmointiympäristöjä sen virheenetsintään. Se poikkeaa tässä työssä kehitetystä metodologiasta siten, että siinä käytetään palvelinkoodin editointiin webhotellin tarjoamaa editointitoimintoa ja sen virheenetsintään käytetään palvelimelta ladatun datan selaimen konsoliin ja näytölle tulostusta sekä kehittäjän tietokoneelle luotavan lokaalipalvelimen konsoliin tulostusta. Virheenetsintään käytetään lisäksi webhotellin tarjoamaa mahdollisuutta seurata tietokantaan tapahtumia muutoksia sovellusta käytettäessä sekä käyttöliittymän havainnointia eri tilanteissa. SPA (*Single Page Application*) – yhden sivun sovellus on malli, jolla viitataan tapaan toteuttaa websovelluksen käyttöliittymä. Kehitettävän järjestelmän kannalta parhaaksi arvioitiin tutkimus, jossa toteutetaan juuri SPA-mallia noudattava työnhallintajärjestelmän prototyyppi pieniksi ja keskisuuriksi luokiteltaville yrityksille. Se on toteutettu moduulirakenteisena AngularJS-sovelluskehystä käyttäen ja se koostuu muun muassa laskutuksesta ja varastonhallinnasta vastaavista moduuleista. Se poikkeaa tässä työssä kehitettävästä järjestelmästä ensinnäkin siten, että kehitystyössä päätettiin olla käyttämättä sovelluskehyksiä käyttöliittymän kehittämisessä. Lisäksi järjestelmää ei suunniteltu vastaavalla tavalla modulaariseksi, vaikka mahdollinen tarve modulaaristen valmisosien käyttöön tiedostettiinkin. Toimeksiantajan päätyminen valmisohjelmiston valintaan kesken projektin aiheutti tarpeen lisätä työhön projektin itsearviointi mukaan lukien työtulokset sekä kehittämisessä käytetyt työkalut. Työnhallintajärjestelmä kehitettiin siitä huolimatta käyttöönottovaihetta vaille valmiiksi. Projektissa toteutuneiden prosessien arviointikriteerinä käytettiin standardin ISO/IEC 29110 tarjoamaa ohjeistusta, joka sisältää mallit projektinhallinta- ja sovelluskehitysprosesseille mukaan lukien tehtävät niiden toteuttamiseksi. Prosessien arviointi perustuu ohjeistuksen määrittelemien tehtävien vertaamiseen toteutumaan. Työtuloksia eli toteutettua sovellusta arvioidaan muun muassa laajennettavuuden osalta. Kehittämiseen käytettäviä työkaluja arvioidaan niiden käytettävyyden kannalta. Työn tulos on käyttöönottoa vaille valmis työnhallintajärjestelmä sekä edellä kuvattu itsearviointi. Itsearvioinnin tuloksien perusteella tehdyistä johtopäätöksistä keskeisimpiä ovat kehitystyössä toteutettava ohjelmiston jako komponenteiksi, jonka todettiin edistävän muun muassa sovelluksen ylläpidettävyyttä ja testattavuutta. Komponenttijakoa taas pääteltiin edesauttavan sovelluskehyksien käyttö. Kehittämismetodologian suhteen todettiin, että mahdollisuutta verkossa toimivien koodieditorien ja ohjelmointiympäristöjen käyttöä suoraan webhotelliin kehitettävän palvelinsovelluksen kehittämisessä vastaavasti kuin löydetyssä tutkimuksessa kannattaa selvittää.

Avainsanat: ERP, yhden sivun sovellus, development of remote web application, ISO/IEC 29110

# ABSTRACT

Matti Kinnunen: ERP for small enterprise: Development and self-assessment
Master of Science thesis
Tampere University
Degree Programme in Management and Information Technology
May 2021

---

ERP systems (Enterprise Resource Planning) are IT based enterprise systems which can be used to integrate various processes and data processing into a single entity. Work management systems are classified as ERP systems as well but in this master thesis their purpose of use and scope were interpreted to be reduced in comparison with full scale ERP systems. The main goal of this thesis is to develop a base version of a work management system which will be based on the requirements and plans of a customer. The system is required to have a browser user interface and the base version is initially planned to be expanded by invoicing and inventory management functions after this project. Based on ISO/IEC 29110 standard the customer of the system is classified as a very small entity. Its business is to maintain and repair the sewage systems of individual homes and cottages. To achieve the goals of this work a methodology which enables the accomplishment of the project remotely will be developed. The methodology must take account the fact that the customer has no ability to inspect and accept the development stages of the system by using source code only. A study which proposes a method for developing web applications on a remote server was evaluated as best to develop the methodology. The method includes the use of a web editor for editing files on the server and the use of a debugger of a web-based IDE for debugging a program on a remote server. It differs from the method which was developed for this project in such a way that an editing function provided by web hosting service was decided to use for editing files on the server. As to debugging of the server code, it was decided to use a console of a web browser to print and observe data downloaded from the server, printing the data on display as well and if necessary, to use a local server. In addition, it was planned to utilize the observing of database tables and user interface changes during the use and tests of the system. With regard to how the browser-based ERPs for very small enterprise should be made it was found one study where a prototype of SPA-based (Single Page Application) ERP system for small and medium-sized enterprises was developed. It consists of accounting and inventory management modules and it has developed by using AngularJS framework. It differs from the system developed in this work because there were no user interface frameworks used. In addition, the system was planned to develop without the module structure, but it was realized there could be some kind of need to ready-made modules. The customer decided to purchase a ready-made ERP system and not to utilize the system which was under development. That caused a need to include self-assessment part in this work. The development of the system except the deployment stage was finished anyway. It was decided to use ISO/IEC 29110 standard based processes as a criterion of processes which were completed in the development project. Those processes include tasks which are compared with the processes completed in the project. When it comes to the assessment of the system itself, inter alia the expandability of the system is assessed. Also, usabilities of development tools are assessed. Therefore, one of the results of this thesis is a work management system whose deployment stage is omitted. The other result is the self-assessment described above. The main conclusion based on result of the self-assessment is that software must be designed so that it consists of components. That kind of structure was founded to increase maintainability and testability. In addition, the use of web frameworks was concluded to help the identification and design of software components. The recommendation of this thesis is also to find out if the use of a web editor for editing files on the server and the use of a debugger of a web-based IDE for debugging a program on a remote server is possibly on servers provided by web hosting services respectively it was used in the study in which a method for developing web applications on a remote server was proposed.

Keywords: ERP, SPA, development of remote web application, ISO/IEC 29110

# ALKUSANAT

Löysin aiheen tähän diplomityöhön Kaveko Oy:n Aarresaari-verkostoon jättämän ilmoituksen perusteella. Itse ohjelmistoprojektin toteutin opiskelijavetoisesti etänä lokakuun 2019 ja elokuun 2020 välisenä aikana. Diplomityönohjaaja vaihtui marraskuussa 2019 ensimmäisen siirryttyä toisen yliopiston palvelukseen ja yliopistonlehtori Timo Mäkisen siirryttyä diplomityönohjaajaksi. Sain häneltä asiantuntevaa tukea varsinkin työn tutkimusnäkökulman kannalta – siitä kiitos hänelle. Kiitän lisäksi Kaveko Oy:tä opettavaisesta haasteesta ja erityisesti toimitusjohtaja Joonas Korkiakoskea, joka toimi työn ohjaajana yrityksen osalta.

Varkaudessa 14.05.2021

Matti Kinnunen

# SISÄLLYSLUETTELO

# KUVALUETTELO

# TAULUKKOLUETTELO

# TERMIT JA MERKINNÄT

| | |
|---|---|
| Agile Solo | on mm. Scrumiin perustuva itsenäisen kehittäjän menetelmä |
| AngularJS | on JavaScript-ohjelmistokehys, joka mahdollistaa muun muassa uudelleen käytettävät ohjelmistokomponentit |
| cPanel | on webhotelleissa käytettävä hallintapaneeli, joka sisältää muun muassa tiedostojen ja tietokannan hallinnan |
| CRUD | lyhenne sanoista *Create, Read, Update and Delete* jotka viittaavat pitkäaikaisessa tiedon varastoinnissa kuten tietokannoissa käytettäviin perusfunktioihin |
| CSS | lyhenne sanoista *Cascading Style Sheets* ja se on websivujen tyylimäärittelyihin tarkoitettu ohjelmointikieli |
| dhtmlx | ohjelmakirjasto, joka tarjoaa yhteensä 28 muokattavissa olevaa ohjelmistomoduulia erilaisiin tarpeisiin |
| ERP | *Enterprise resource planning* on tyypillisesti yritysohjelmisto, josta käytetään suomenkielistä ilmaisua toiminnanohjausjärjestelmä |
| event | Scheduler-moduulin käyttämä tietorakenne, jota käytetään kalenteritapahtumien näyttämiseen kalenterinäkymässä |
| File Manager | on cPanel-hallintapaneelin sisältämä tiedostonhallintassovellus, johon kuuluu myös perustoiminnot koodin editointiin |
| Gantt-kaavio | projektinhallinnassa käytetty janakaavio |
| GanttProject | Gantt-kaavioiden käyttöä tukeva sovellus |
| Gzip | on tiedoston pakkaustekniikka |
| HTML | *Hypertext Markup Language* on merkintäkieli, jolla kirjoitetaan varsinkin internetsivujen staattiset osat |
| IDE | *Integrated Development Environment* on tyypillisesti sovellus, joka tarjoaa ohjelmointiympäristön sisältäen muun muassa virheenetsintätyökalut |
| ISO/IEC 29110 | on standardi ohjelmistotuotantoa harjoittavien pienorganisaatioiden tuotteiden laadun ja prosessien parantamiseen |
| ISO/IEC 33020 | on standardi, joka sisältää viitekehyksen prosessien kyvykkyyden arviointiin |
| JQuery | kaikissa selaimissa toimiva avoimen lähdekoodin JavaScript-kirjasto |
| JSON | *JavaScript Object Notation* on tiedostomuoto, jota käytetään tiedonvälitykseen – esimerkiksi selaimen ja palvelimen välillä |
| MPA | *Multi page application* on monisivuinen websovellus ja sillä viitataan perinteiseen tapaan toteuttaa internetsivut |
| MVVM | *Model-View-ViewModel* on websovellusten selainpuolen sovelluksen arkkitehtuuri |
| Node.js | alusta JavaScript-koodin tulkkaukseen ja sovellusten ajamista varten |
| nodeJS | viittaa vastaavaan ilmaisuun kuin Node.js |
| N-P-L-F | on standardissa ISO/IEC 33020 määritelty nelitasoinen asteikko prosessiattribuuttien luokitusta varten |
| package.json | on Node.js-palvelinsovelluksen käyttämä konfigurointitiedosto |
| pk | pieni ja keskisuuri, jolla viitataan yrityksen kokoluokkaan |
| PSP | *Personal Software Process* on henkilökohtainen ohjelmistokehitysprosessi, joka mahdollistaa sovelluskehittäjän ammattitaidon itsearvioinnin ja parantamisen |
| PXP | *Personal Extreme Programming* on itsenäisen kehittäjän menetelmä, jossa on yhdistetty ketteriin menetelmiin luokiteltavan |

| | |
|---|---|
| | XP:n ja henkilökohtaisen ohjelmistokehitysprosessin (engl. *Personal Software Process* eli PSP) käytäntöjä |
| ready | JQuery-kirjaston funktio, jonka käyttötarkoitus on varmistaa, ettei toteutettava sovelluslogiikka yritä suorittaa toimintoja ennen kuin sivusto on siihen valmis |
| REST | *Representational state transfer* on ohjelmointirajapintojen toteuttamiseen tarkoitettu arkkitehtuurimalli, joka määrittelee sovellukset niiden komponenttien, resurssien sekä niiden yhteyksien kautta |
| SC | Softwate Change on SIP- ohjelmistokehitysmallin käyttämä käsite ohjelmamuutoksesta |
| Scheduler | on dhtmlx-kirjaston tarjoama kalenterimoduuli |
| Scrum | on ketteriin menetelmiin luokiteltava ohjelmistokehitysprosessi |
| Setup Node.js | Net9 Small-webhotellin käyttämään cPanel-hallintapaneeliin integroitu sovellus Node.js-palvelinsovellusten luomista, muokkausta ja suorittamista varten |
| SIP | *Solo Iterative Process* on ohjelmamuutokseen perustuva ohjelmistokehitysmalli |
| SPA | *Single Page Application*, on eräänlainen websovellusrakenne josta käytetään suomenkielistä ilmaisua yhden sivun sovellus |
| Webpack | on tiedoston niputustekniikka |
| XP | on ketteriin menetelmiin luokiteltava ohjelmistokehitysprosessi |
| zip | on tiedostomuoto, jota käytetään tiedon pakkaukseen |

# 1. JOHDANTO

Työnhallintajärjestelmät luokitellaan tässä työssä toiminnanohjausjärjestelmiksi (*ERP*, *Enterprise resource planning*). Toiminnanohjausjärjestelmiä käyttävät pääasiassa suuryritykset, mutta niitä on saatavilla myös pienille ja keskisuurille yrityksille. Varsinkin suuryrityksille suunnitellut toiminnanohjausjärjestelmät ovat tyypillisesti alustoja, joihin voi integroida hyvin monenlaisia prosesseja ja niitä tyypillisesti lisätään järjestelmään moduuleina. Työn perusteella toiminnanohjausjärjestelmien kehittäminen alusta asti ilman, että se perustuu edellä mainittuun valmiiseen alustaan perustuvaan rakenteeseen, on melko harvinaista. Toisaalta tämän työn tavoitteena on kehittää työnhallintajärjestelmä juuri mainittuun harvinaisuuteen perustuen. Tämän taustalla on toimeksiantajan näkemys, jonka mukaan saatavilla olevat toiminnanohjausjärjestelmät eivät ole sille sopivia. Sen johdosta se päätti aloittaa omiin tarpeisiin räätälöidyn, verkkoselaimessa toimivan työnhallintajärjestelmän kehittämisen. Toimeksiantaja antoi projektin alussa vaatimuksia, joihin perustuen sovellus on tarkoitus toteuttaa. Projektin alussa sovittiin myös alustavasti, että työnhallintajärjestelmää laajennetaan myöhemmän erillisen projektin puitteissa ainakin laskutuksen ja varastonhallinnan vaatimilla ominaisuuksilla ja toiminnallisuuksilla.

Toimeksiantaja on jätevesijärjestelmien huoltoja ja korjauksia tekevä yritys, jonka asiakkaat sijaitsevat eri puolilla Suomea. Sen osaamisaluetta ovat muun muassa panos- ja pienpuhdistamojen huollot ja korjaukset. Henkilöstömäärältään tämä kasvuun pyrkivä yritys koostuu kahdesta henkilöstä, joista molemmat osallistuvat yrityksen vastaanottamien toimeksiantojen toisin sanoen asiakkaiden tilaamien toimenpiteiden suorittamiseen.

Projekti on tarkoitus toteuttaa toimeksiantajayritykseen nähden lähinnä etänä, sillä sen molemmat toimipisteet sijaitsevat diplomityöntekijän asuinpaikkaan nähden yli 300 kilometrin ajomatkan päässä. Työ aloitetaan kirjallisuuskatsauksella. Sen avulla pyritään etsimään keinoja, kuinka selaimessa toimiva, pienelle yritykselle kehitettävä työnhallintajärjestelmä kannattaa toteuttaa etänä. Sen jälkeen luodaan metodologia, jota noudattaen vaatimustenmukainen työnhallintajärjestelmä pyritään rakentamaan. Kehittämismetodologian kannalta tärkeimmäksi hakutulokseksi osoittautui tutkimus [17], jonka tuloksena ehdotetaan metodologiaa websovelluksen kehittämiseksi suoraan palvelimelle kehittäjän tietokoneelta käsin. Itse kehitettävän järjestelmän kannalta tärkeimmäksi hakutulokseksi valikoitui ilman valmista alustaa hyödyntävä tutkimus [7], jossa kehitettiin toi-

minnanohjausjärjestelmän prototyyppi pieniin ja keskisuuriin luokiteltaville yrityksille. Kyseessä on niin sanottua yhden sivun sovellus (*Single Page Application - SPA*) – toteutustapaa hyödyntävä moduulirakenteinen sovellus. SPA viittaa websovellukseen, jossa selainpuolen sovellus lataa kaikki sivuston osat ensimmäisen verkkopyynnön yhteydessä, jonka jälkeen selaimen ja palvelimen välinen liikenne koostuu vain datasta [7]. Kirjallisuuskatsauksen muihin lähtökohtiin liittyen lopullisiksi tietolähteiksi valikoituivat itsenäisen kehittäjän menetelmiin keskittyvä pro gradu – tutkielma [14] sekä lähde [13] prototypointia soveltavan metodologian kehittämiseksi.

Ohjelmistoprojekti toteutetaan kehitettyä metodologiaa käyttäen. Projektin tulos on selaimessa toimiva työnhallintajärjestelmän perusversio, joka sisältää muun muassa asiakasrekisterin ja toimenpiteiden selauksen haku- ja muokkaustoiminnallisuuksineen, uuden toimenpiteen luonnin yhteydessä työntekijälle lähetettävine tekstiviesteineen sekä kalenterin. Projektin valmistuttua se arvioidaan. Projektiarviointi kattaa paitsi projektin tuottaman ohjelmistotuotteen, myös sen toteuttamiseen käytetyt prosessit ja käytetyt työkalut. Ohjelmistotuote arvioidaan muun muassa sen laajennettavuuden kannalta. Se rajataan vain itse sovelluksen laajennettavuuteen eikä siihen integroitavissa olevien moduulien saatavuus kuulu työn aihepiiriin. Arvioinnin lähtökohtana on toimeksiantajan päätös perua kehitteillä olleen sovelluksen hyödyntäminen toiminnassaan ja valita sen sijaan valmisohjelmisto. Siitä johtuen, myös arviointi päätettiin sisällyttää työn aihepiiriin. Työnhallintajärjestelmä kehitettiin siitä huolimatta käyttöönottovaihetta vaille valmiiksi. Ohjelmistoprojektin arviointikriteerinä päätettiin käyttää pienorganisaatioiden tuottamien tuotteiden laadun ja prosessien parantamiseen suunniteltua standardia *ISO/IEC 29110* [12] niin, että sen kuvaamia prosesseja verrataan toteutetun ohjelmistoprojektin vastaaviin. Standardi *ISO/IEC 33020* sisältää viitekehyksen prosessien kyvykkyyden arviointiin ja siihen kuuluvaa *N-P-L-F*-asteikkoa käytetään paitsi prosessien arviointiin, myös työtulosten ja työkalujen tarkasteluun [11]. Projektin arvioinnin tarkoituksena on mainittujen ohjelmistokehitykseen liittyvien osa-alueiden parantaminen itsenäisen kehittäjän kannalta.

## 1.1 Tavoitteet ja sisältö

Työn päätavoite on työnhallintajärjestelmän kehittäminen toimeksiantajan vaatimuksiin perustuen. Diplomityössä tehtävän kirjallisuuskatsauksen ohjenuorana on käytetty Barbara Kitchenhamin metodia. Projektin tavoitteiden saavuttamiseksi luodun kehittämismetodologian osana käytetään V. Basilin kehittämää The engineering method - mallia itse ohjelmistokehityksen osalta. Projektiarvioinnissa on kyse tapaustutkimuksesta, johon liittyen esimerkiksi Robert K. Yin teoksessaan Case Study Research and Applications kuvaa menetelmät.

Työn punaiseksi langaksi muotoiltiin seuraavat tutkimukselliset kysymykset:

- TK1: Miten etänä toteutettava websovellusprojekti kannattaa toteuttaa?
- TK2: Kuinka alusta asti kehitettävä, selaimessa toimiva työnhallintajärjestelmä kannattaa toteuttaa pienelle yritykselle?
- TK3: Mitkä ovat toteutetun ohjelmistoprojektin heikkoudet ja kuinka niiden suhteen olisi pitänyt toimia?

Järjestelmän kehittämisen tarkoitus on helpottaa toimeksiantajan liiketoimintaprosesseja. Koska kyseessä on etänä toteutettava projekti ja itse kehitystyön reunaehtona alusta asti kehitettävä työnhallintajärjestelmä, tutkimuksellisiin kysymyksiin TK1 ja TK2 on tarkoitus etsiä vastauksia. Tutkimukselliseen kysymykseen TK3 etsitään vastauksia, jotta tämän projektin ohjelmistokehitykseen liittyvien osa-alueiden parantaminen itsenäisen kehittäjän kannalta olisi mahdollista.

Työn toisessa luvussa tehdään kirjallisuuskatsaus vastausten löytämiseksi tutkimuksellisiin kysymyksiin TK1 ja TK2. Myös katsauksen tuloksena löydetty referenssijärjestelmä ja sen yhteys tavoiteltuun järjestelmään käsitellään kyseisessä luvussa. Viidennessä luvussa arvioidaan ohjelmistoprojekti, sen tuloksena tuotettava järjestelmä ja kehitystyössä käytettävät työkalut. Kyseisessä luvussa käytettävä kriteeristö ja menetelmät esitellään luvussa kolme. Viidennessä luvussa tehtävä projektiarviointi antaa vastauksen tutkimukselliseen kysymykseen TK3. Neljännessä luvussa käydään läpi tavoitellun järjestelmän tuottava ohjelmistoprojekti.

Liitteessä A esitellään toimeksiantajan asettamat vaatimukset kehitettävälle järjestelmälle sekä niiden mallinnuksen. Liite B sisältää pääosan kehitettyä järjestelmää varten tehdystä vaatimusmäärittelystä sekä toiminnalliseen määrittelyn, jonka toteutuksessa on käytetty standardia IEEE 830 ohjenuorana. Liite A ja osin myös liite B kuvaa toimeksiantajan näkökulman tutkimukselliseen kysymykseen TK2. Liitteeseen C on koottu järjestelmän tärkeimmät funktiokutsut ja tilatiedot näkymien välillä navigoitaessa ja se on luotu helpottamaan järjestelmän lähdekoodin omaksumista mahdollisissa jatkokehitys- tai muokkaustilanteissa. Kyseinen liite kuvaa tutkimuksellisen kysymyksen TK2 vastausta lähdekoodin osalta. Liite D sisältää järjestelmän käyttöönottovaihetta edeltävät testitapaukset ja niiden toteutuspäivämäärät. Sitä käytetään osana kysymykseen TK3 liittyvää vastausten etsintää.

# 2. LÄHTÖKOHDAT JA KEHITTÄMISMETODOLOGIA

Tämän luvun ensimmäisessä aliluvussa kuvataan ensin työn tausta alkaen toiminnanohjausjärjestelmistä päättyen projektin lähtökohtaan, eli toimeksiantajan tarpeeseen kehittää itselleen räätälöity työnhallintajärjestelmä. Toisessa aliluvussa taustoitetaan ensin tutkimuskysymysten TK1 ja TK2 perusteella tehtävää kirjallisuuskatsausta ja kuvataan hakutuloksien valintakriteerit sekä käytettävät tietolähteet. Kolmanteen alilukuun on koottu katsauksen lopulliset tulokset ja viimeisessä aliluvussa kuvataan kehitetty metodologia sekä sen yhteys kirjallisuuskatsauksen tuloksiin.

Kehittämismetodologian muodostamisen lähtökohta on, että itse ohjelmistoprojekti oli jo käynnissä ennen kuin lopullinen versio metodologiasta oli valmis. Projektin alussa sovelluksen ensimmäisiä versioita visualisoitiinkin toimeksiantajalle lähettämällä tälle *HTML* -sivuja tarvittavine *JavaScript*-tiedostoineen *zip*-muodossa. Lyhenteellä HTML viitataan merkintäkieleen, jolla kirjoitetaan tyypillisesti internetsivujen staattiset osat ja lyhenne zip viittaa tiedon pakkaukseen käytettävään tiedostotyyppiin. Lopullisesta metodologiasta oli silti perusidea olemassa jo ennen projektin ensimmäistä tapaamista, jossa toimeksiantajalle mainittiin mahdollisuudesta kehittää sovellusta suoraan webhotelliin, jonka kautta kehitettäviä ominaisuuksia olisi mahdollista visualisoida.

## 2.1 Työn tausta

Toiminnanohjausjärjestelmät ovat yritysten tietojärjestelmiä, joilla organisaatiot voivat integroida reaaliaikaisesti datan, prosessit ja informaatioteknologian hyödyt yhdeksi ”paketiksi” kattaen sekä yrityksen sisäiset, että ulkoiset arvoketjut. Suurten toimittajien, kuten SAP, Oracle, PeopleSoft, Siebel, ja i2 Corporation toiminnanohjausjärjestelmät koostuvat tyypillisesti sovellusmoduuleista, joita otetaan käyttöön tarpeen mukaan. Moduuleista on mahdollista koota joustavasti toiminnanohjausjärjestelmä hyvin erilaisiin tarpeisiin ja erilaisille toimialoille. Toimittajat tarjoavat usein järjestelmiään käyttöön lisenssimaksua vastaan ja sitä pidetään edullisempana vaihtoehtona kuin kokonaan räätälöidyn järjestelmän kehittäminen yritykselle. Haasteeksi lisenssin hankkivalle yritykselle voi muodostua sopivan prosessikombinaation valinta ja niiden käyttöönotto järjestelmän tarjoamista moduuleista, sillä markkinoilla olevat toiminnanohjausjärjestelmät ovat usein eräänlaisia puolivalmisteita sisältäen muun muassa tauluja ja parametreja, jotka yrityksen ja sen asiakasorganisaatioiden on joka tapauksessa konfiguroitava ja muokattava omiin tarpeisiin sopiviksi sekä mahdollisesti myös integroitava ne jo olemassa oleviin tietojärjestelmiin [22].

Lähteen [22] sisältämän informaation perusteella erilaiset työnhallintajärjestelmät luokitellaan toiminnanohjausjärjestelmiksi. Käsitteitä käytetäänkin tässä työssä rinnakkain. Työnhallintajärjestelmät tulkittiin kuitenkin olevan käsitteenä suppeampi kuin täysimittaiset toiminnanohjausjärjestelmät.

Tilastokeskuksen [25] mukaan ERP-ohjelmisto oli 39 prosentilla yrityksistä vuonna 2017. Erot yritysten suuruuden mukaan ovat sen mukaan suuria, sillä pienimmistä tarkastelluista yrityksistä 25 prosenttia ja suurimmista 82 prosenttia käytti ERP-ohjelmistoa [25].

Myös *pk-yrityksille* on saatavilla vaihtoehtoja. Pk-yrityksillä viitataan pieniin ja keskisuuriin yrityksiin. Eräs niistä on toimeksiantajayrityksen käyttämä Tribuni, joka tosin oli kodinkonehuoltoa tekeville yrityksille suunniteltu versio eikä Tribunista ole saatavilla erillistä versiota jätevedenpuhdistamojen huoltoja ja korjauksia tekeville yrityksille. Lisäksi ainakin SAP tarjoaa pk-yrityksille suunnitteltua versiota. Myös vapaan lähdekoodin vaihtoehtoja löytyy. Niitä ovat ainakin ADempiere, Apache OFBiz ja Odoo.

Toimeksiantajayrityksen aiempi kokemus kokonaan toiselle toimialalle suunnitellun toiminnanohjausjärjestelmän käytöstä oli perustana sille, että sillä oli valmiita ideoita itse kehitettävää järjestelmää varten. Aiemmin käytössä ollut Tribuni-toiminnanohjausjärjestelmä ei sopinut sen tarpeisiin muun muassa siksi, että kodinkonehuoltoja tekevälle pk-yritykselle suunnitellun toiminnanohjausjärjestelmän lomakkeet sisälsivät turhia tekstikenttiä, jonka vuoksi se koettiin käytettävyydeltään sekavaksi.

## 2.2 Aiempien tutkimusten haku ja valinta jatkotarkasteluun

Peruslähtökohtana tälle kirjallisuuskatsaukselle on tutkimuskysymykset TK1 ja TK2 eli katsauksessa keskitytään siihen, kuinka verkkoselaimessa toimivan työnhallintajärjestelmän tuottava yhden hengen ohjelmistoprojekti kannattaa toteuttaa etänä. Diplomityöntekijällä on positiivisia kokemuksia ketterien menetelmien inkrementaalisesta kehitystavasta, jossa ohjelmiston osia tehdään valmiiksi sprintteinä, joiden päätteeksi pyritään saamaan toimeksiantajan hyväksyntä iteraatioissa toteutetuille toiminnoille. Tämä päätettiinkin ottaa lähtökohdaksi kehittämismetodologian muodostamiselle toimeksiantajan ilmaistua projektin ensimmäisessä palaverissa valmiutensa kyseisen toteutustavan käyttöön.

Diplomityöntekijän käsitys SPA-toteutusten hyödyistä websovelluskehityksessä vaikutti päätökseen hyväksyä kirjallisuuskatsauksen haku- ja valintakriteereihin myös SPA-mallia hyödyntävät ERP-toteutukset. SPA-mallia noudattavaa sovellusta oli sitä paitsi ehdotettu alustavaksi ratkaisuksi toimeksiantajalle jo projektin aloitustapaamisessa. Reunaehdoksi todettiin, että toimeksiantajalla ei ole valmiutta verkossa toimivan versionhallin-

tahakemiston kautta tapahtuvaan toteutuneiden toimintojen tarkasteluun eli kyseistä tarvetta varten täytyy löytää tai kehittää tarkoituksenmukainen menetelmä. Lisäksi diplomityöntekijän kiinnostus prototypointia kohtaan päätettiin sisällyttää yhdeksi kirjallisuuskatsauksen lähtökohdista.

Lähtökohtien perusteella muodostettiin ensin hakufraasi, joka on muotoa ”remote web development erp small enterprise”. Koska fraasiin liittyvät lähtökohdat muodostavat melko erityislaatuisen tutkimusalueen, päätettiin mainittu hakufraasi jakaa lisäksi muotoa ”remote web development” ja ”erp small enterprise” oleviin osiin. Ensimmäisen hakufraasin jakaminen kyseisellä tavalla nähtiin tarkoituksenmukaiseksi, sillä molemmat hakufraasit nähtiin käsittävän erilliset, tavoitellut aihepiirit, joita käyttämällä hakutuloksia kannattaa etsiä. SPA-mallia hyödyntävien tutkimusten hakemiseksi muotoiltiin hakufraasi ”ERP single page application spa. Hakutuloksista päätettiin sisällyttää jatkotarkasteluun otsikot, jotka liittyvät parhaiten hakufraaseihin, mutta säännöstä poikettiin siltä osin, että jos otsikko viittasi muulla tavoin tavoiteltuun sisältöön, niin valinta tehtiin hakutuloksen sisällön tarkastelun jälkeen. Pienille organisaatioille tarkoitetut työnhallintajärjestelmät tiedettiin suurempien yritysten vastaavia vähemmän edustetuiksi, joten hakutuloksiin päätettiin sisällyttää myös lupaavimmat osumat ylipäänsä toiminnanohjausjärjestelmien kehittämiseen liittyen. Poikkeukseksi hyväksyttiin myös yleensäkin etänä tapahtuvaan ohjelmistokehitykseen liittyvät hakutulokset. Pääperiaate jatkotarkasteluun valinnalle oli kuitenkin otsikon sisältämän informaation vastaavuus hakufraasiin nähden.

Tietokantojen hakutoiminnallisuus päätettiin jättää niiden oletusasetukseen paitsi hakutulosten järjestelyn osalta, joka asetettiin kaikkien tietokantojen tapauksissa muotoon ”Sort by relevance” tai vastaava. Se olikin oletuksena suurimmassa osassa tietokantoja. Hakutuloksista päätettiin tehdä valinta ensimmäisten 50 hakutuloksen joukosta. Edellä mainittujen hakuehtojen perusteella tehtyjen hakujen tuloksista valittiin jatkotarkasteluun yhteensä 135 hakutulosta. Hakulähteiksi valittiin taulukkoon 1 kootut tietokannat.

***Taulukko 1. Kirjallisuuskatsauksessa käytetyt hakulähteet.***

| Tietokanta |
|---|
| Aaltodoc, Master's theses |
| Academic Search Ultimate (Ebsco) |
| ACM Digital Library. Association for Computing Machinery |
| Andor |
| Applied Science & Technology Source (Ebsco) |
| Communication & Mass Media Complete (Ebsco) |
| Computer Science Database (ProQuest) |
| Emerald |
| IEEE/IET Electronic Library (IEL) / IEL |
| Library, Information Science & Technology Abstracts (LISTA) (Ebsco) |
| Library & Information Science Collection (ProQuest) |
| ScienceDirect (Elsevier) |
| Scopus (Elsevier) |
| SpringerLink |
| Trepo |
| Web of Science |

Sen jälkeen jatkotarkasteluun päätyneistä hakutuloksista pyrittiin rajaamaan ulos kaikki hakutulokset, jotka eivät sisällä konkretiaa kirjallisuuskatsauksen lähtökohtiin nähden. Tällöin hakutuloksista rajattiin pois esimerkiksi osumat, jotka tarkemmassa tarkastelussa osoittautuivat toiminnanohjausjärjestelmien kehittämisen sijaan niiden käyttöönottoon liittyviksi. Hakutuloksista rajattiin pois myös tulokset, joista huomattiin, että niiden koko teksti ei ole saatavilla. Myöhemmin tehtiin vielä erilliset haut yksin suoritettavaan ohjelmistokehitykseen liittyen fraasilla ”solo agile” ja prototypointiin liittyen fraasilla ”prototyping model in software engineering” vastaavaa iterointia hyödyntävää hakumenetelmää käyttäen kuin edellisessä haussa. Viimeksi mainittu hakufraasi perustuu ennen kirjalli-

suuskatsausta Google-hakukoneella tehtyyn alustavaan tiedonetsintään. Jatkotarkastelun tuloksena kirjallisuuskatsauksen lopullisiksi valinnoiksi valikoitui lopulta 4 hakutulosta, jotka esitellään seuraavassa aliluvussa.

## 2.3 Kehittämismetodologian kannalta merkittävimmät hakutulokset

### KÄHKÖNEN (2014)

Pro gradu – tutkielmassa [14] pyritään etsimään tietojenkäsittelyn tietokannoista ja verkosta tutkimusartikkeleita, jotka käsittelevät itsenäisen ohjelmistokehittäjän malleja esitteleviä tutkimuksia. Hän määrittelee itsenäiseksi kehittäjäksi ryhmän jäsenen, jolla on paljon itsenäistä työtä, mutta koodi, joka tuotetaan, on yhteinen [14, s. 4]. Toinen esimerkki on ohjelmistoalan yksityisyrittäjä ja kolmas ohjelmistoprojekteja itsenäisesti tekevä ohjelmoija ”joka kehittää tuotetta ilman projektia tilannutta asiakasta joko itselle tai markkinoille myyntiin tai avoimena lähdekoodina ilmaiseksi.” [14, s. 4]. Työn lopuksi valituista ja analysoiduista tutkimuksista kootaan synteesi eli tietojen yhdistäminen [14, s. 22]. Tutkielman mukaan soolomalleja on kolmenlaisia: Henkilökohtaisen ohjelmistoprosessin ja ketterän menetelmän yhdistävät, yhtä ketterää menetelmää käyttävät tai useampaa yhdistävät sekä näistä erillinen menetelmä *Solo Iterative Process* [14, s. 2]. Kyseisissä menetelmissä toistuvat itsenäiselle kehittäjälle hyväksi havaittuina käytäntöinä lähteen mukaan iteratiivisuus, vaatimussuunnittelu asiakkaan kanssa, visualisointi, tuotteen kehitysjono, töiden kohtuullistaminen, yksikkötestaukset ja versionhallinta [14, s. 4].

Kähkönen esittelee työssään 3 itsenäisen kehittäjän menetelmää [14, s. 34], joita ovat *PXP* (*Personal Extreme Programming*) [14, s. 34], *Agile Solo* [14, s. 37] ja Solo Iterative Process [14, s. 38]. PXP:ssä on yhdistetty ketteriin menetelmiin luokiteltavan *XP*:n ja henkilökohtaisen ohjelmistokehitysprosessin (engl. *Personal Software Process* eli *PSP*) käytäntöjä [1, katso 14, s. 34-35].

PSP mahdollistaa sovelluskehittäjän ammattitaidon itsearvioinnin ja parantamisen. Menetelmässä kaikki työt ensin suunnitellaan toimintaohjeiden mukaisesti. Kuhunkin työvaiheeseen kulunut aika mitataan ja kirjataan ylös – ohjelmassa havaittujen puutteiden, niiden korjaustapojen sekä valmistuneiden tuotteiden lisäksi. Laadukkaiden tuotteiden tuottamiseksi tuotteen laatua suunnitellaan, mitataan ja seurataan. Laatuun siis panostetaan alusta asti. Lopuksi kehittäjä analysoi tulokset ja käyttää analysoinnin tuloksia itsensä kehittämiseen. [10, katso 14, s. 14-15].

PXP:ssä XP:n ryhmäkäytännöt on joko poistettu tai muutettu, mutta menetelmä huomioi koodin yhteisomistajuuden. Siinä otetaan huomioon myös asiakkaan mahdollinen kykenemättömyys olemaan läsnä yksittäisen kehittäjän projektissa, jonka perusteella PXP:ssä ehdotetaan puheluita ja viestittelyä yhteydenpitoon asiakkaan kanssa. Muut käytännöt pidetään sellaisenaan. PSP:stä on otettu menetelmään mukaan sen toiminta-ohjeet. [1, katso 14, s. 34].

*SIP* (*Solo Iterative Process)* on lähteen [14] mukaan Rajlichin [21] kehittämä ohjelmistokehitysmalli, joka perustuu ohjelmamuutokseen (engl. *Software Change*, *SC*). Se koostuu eri vaiheista, joiden tuloksena saadaan aikaan muutos ohjelmaan. SIP:ssä toistetaan ohjelmamuutoksia, eli se on iteratiivinen malli. [19, katso 14, s. 38].

SIP-iteraatiossa on seuraavat viisi vaihetta [19, katso 14, s. 38]:

1. Hankitaan, analysoidaan ja priorisoidaan vaatimukset tuotteen kehitysjonoksi.
2. Valitaan toteutettavat vaatimukset ja tehdään niistä iteraation kehitysjono (engl. iteration backlog).
3. Toteutetaan muutokset ohjelmamuutosprosessina (SC) vaatimus kerrallaan.
4. Tallennetaan koodi välillä perushaaraan.
5. Testataan ohjelma ja tehdään julkaisu ennen seuraavaa iteraatiota.

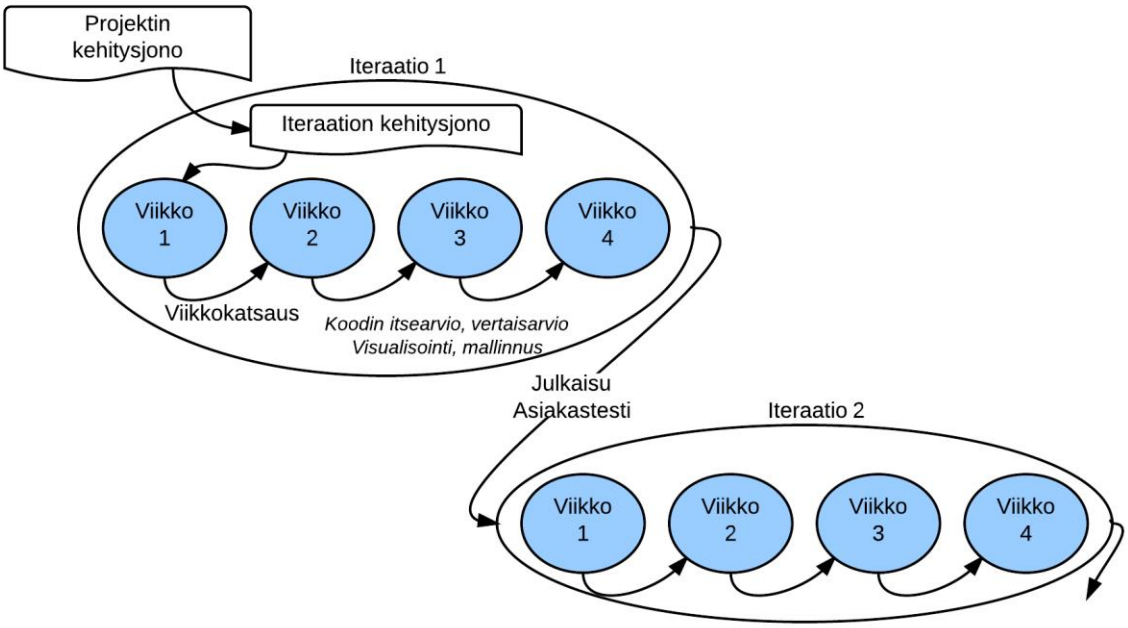
Kähkösen [14, s. 37] löytämän tutkimuksen [18] mukaan tavoitteena Agile Solo-menetelmän kehittämisessä on ollut tehdä ”ketterään julistukseen ja muihin ketteriin malleihin pohjautuva yksittäisen kehittäjän menetelmä.” *Scrumista*, joka sekin on ketteriin menetelmiin luokiteltava ohjelmistokehitysprosessi, Agile Soloon on otettu neljän viikon sprintit, jotka menetelmässä toteutetaan siten, että joka viikko pidetään asiakkaan kanssa esittely jatkuvan palautteen saamiseksi. Esittelyssä muun muassa päivitetään tehtävien tärkeysjärjestystä. Neljännen viikon lopuksi eli sprintin lopussa tuotos julkaistaan ja asiakas testaa sen. Iteraatioita suunniteltaessa tehdään priorisoinnin lisäksi aika-arvio ja sprintin lopussa verrataan sitä todellisuudessa kuluneeseen aikaan. Tämä mahdollistaa jatkossa tarkemmat aika-arviot.

Projektinhallintaan on valittavissa ottaako XP:n käyttäjätarinat vai Scrum-tyyppisen kehitysjonon. Ideana on kuitenkin pitää projektinhallinta mahdollisimman yksinkertaisena ja käyttää sitä joustaen. Agile Soloon kuuluu myös työn visualisointi, jonka toteutuksen voi valita itse. Tällä mahdollistetaan se, että kehittäjä pysyy selvillä työn vaiheista ja voi näyttää niitä myös asiakkaalle. Mallinnusta neuvotaan tekemään vain sen verran kuin se

on tarkoituksenmukaista; sen tarkoitus on helpottaa viestintää asiakkaan kanssa. Ketterissä menetelmissä ilmenevä ohjelman taipumus muuttua voimakkaasti pakottaa mallinnuksen pitämiseen yksinkertaisena. Mallin kehittäjä piti parhaana vaihtoehtona kielellistä keskustelua, koska mallinnuksen käyttö voi hänen mukaansa vaatia asiakkaalta taustatietoa ohjelmistokehityksestä. [18, katso 14, s. 37].

Itse kehitysvaiheessa menetelmä suosii XP:n mukaista testivetoista kehitystä. Koodin arvioinnissa käytetään vertais- ja itsearviointia. Agile Solon kehittäjä ehdottaa, että koodi näytettäisiin kaksi kertaa viikossa toiselle henkilölle arviointia varten. [18, katso 14, s. 37].

Kähkösen [14, s. 37] mukaan tällaista mahdollisuutta ei itsenäisellä kehittäjällä aina kuitenkaan ole. Agile Solo on kiteytetty kuvaan 1.




***Kuva 1.*** *Agile Solo -malli [14].*

## GUNAWAN & KOSALA (2020)

Tutkimuksessa [7] etsitään ratkaisua Indonesian pieniin ja keskisuuriin luokiteltavien yritysten liiketoimintaprosessien hallintaan. Saatavilla olisi tutkimuksen mukaan valmiita toiminnanohjausjärjestelmiä mutta niiden hinnat ovat olleet suhteellisen kalliita Indonesian pienille ja keskisuurille yrityksille. Ongelman ratkaisuksi tutkimuksessa nähdään kohtuuhintainen, etäkäyttömahdollisuudella varustettu toiminnanohjausjärjestelmä, joka

lisäksi käyttää saatavilla olevia pilvipohjaisia toteutuksia vähemmän verkon kaistanleveyttä. Tutkimuksen tavoitteena on analysoida kuvattua ongelmaa ja sen tuloksiin perustuen toteuttaa prototyyppi, joka vastaa ongelmaan. Analyysin yhteenvetona todetaan, että suurin ongelma pienten ja keskisuurten yritysten toiminnanohjaukseen liittyvien ohjelmistojen suhteen on se, että saatavilla ei ole sopivia ERP-ohjelmistoja erityisesti Jakartan vähittäis- ja tukkukaupan alaa edustaville yrityksille. Tutkimuksessa päädytään ehdottamaan edellä kuvattujen ongelmien ratkaisuksi pilvipohjaista järjestelmää, jossa keskitytään kattamaan vähittäis- ja tukkukaupan kannalta olennaiset perusominaisuudet, kuten varastonhallinta, osto- ja myyntitapahtumat sekä erilaisten liiketoimintaan liittyvien raporttien generointi. Analysointiin perustuen tutkimuksessa päädytään toteuttamaan prototyyppi SPA-toteutuksena [7, s. 5]. SPA soveltuu sen mukaan hyvin verkkoyhteyksistä johtuvien hintapaineiden ratkaisuksi.

Samasta syystä tutkimuksessa päätettiin hyödyntää myös tiedoston pakkaus- ja niputustekniikoita. Tiedoston pakkaamista voidaan hyödyntää verkon yli lähetettävän datan minimoimiseen. *Gzip* mahdollistaa websovellusten käyttää tiedoston pakkaamista HTTP-tiedonsiirroissa. Moduulien niputusta käytetään työkaluna useiden resurssitiedostojen niputtamiseksi yhdeksi tai useammaksi osaksi. *Webpack* mahdollistaa erityyppisten moduulien niputtamisen sisältäen JavaScript-, CSS(*Cascading Style Sheets*) - ja kuvatiedostot käyttämällä kullekin erityistä lataajaa. Niputus parantaa asiakas- ja palvelinpään välisen kommunikointiprosessin tehokkuutta. [7, s. 8].

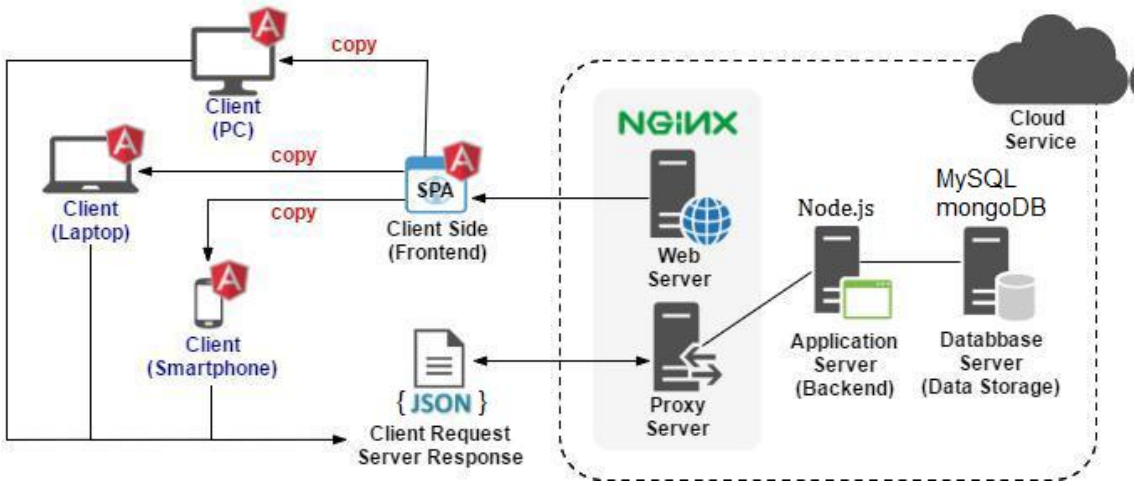
Itse toteutusprojektissa tutkimuksen tekijä kehittää sovelluksen mainituilla teknologioilla niin, että se koostuu kahdesta rakenneosasta; palvelinpuolen *nodeJS*:stä ja selainpuolen *AngularJS*:stä [7, s. 11]. Käsitteellä nodeJS tulkittiin viitattavan *Node.js*-alustaan. AngularJS on websovelluskehys, joka mahdollistaa muun muassa uudelleenkäytettävien ohjelmistokomponenttien käytön [2]. Kirjoittajan fokus sovelluksen kehittämisessä on SPA-malli, joka edellyttää selainpuolen sovelluksen lataavan kaikki sivuston osat ensimmäisen verkkopyynnön yhteydessä ja että selainsovellus kykenee suoriutumaan useista tehtävistä itsenäisesti [7, s. 5]. Kommunikointi, joka toteutuu palvelimen ja selaimen välillä on tarkoitus sisältää dataa, joka on *JSON*-muodossa ja johon pääsee käsiksi *REST* (*Representational state transfer*) -rajapinnan kautta [7, s. 5]. JSON (*JavaScript Object Notation*) on syntaksi datan varastointiin ja siirtoon [9]. Käsitteellä REST viitataan ohjelmointirajapintojen toteuttamiseen tarkoitettuun arkkitehtuurimalliin, joka määrittelee sovellukset niiden komponenttien, resurssien sekä niiden yhteyksien kautta [20].

Verrattuna perinteiseen monisivuiseen sovellukseen (*MPA Multi page application*), joka lataa kaikki sivuun liittyvät resurssit aina sivuun liittyvän verkkopyynnön yhteydessä,

SPA-mallissa ladataan vain tarvittava data, jonka ansiosta se vähentää selaimen ja palvelimen välistä tietoliikennettä [7, s. 5]. Yhden sivun sovellus siis lataa kaikki resurssit sivuston ensimmäisellä latauskerralla mukaan lukien HTML-sivu, JavaScript- sekä sivuston tyylittelyn määrittelevät CSS-tiedostot, joista kaikista sivuston ensimmäinen näkymä luodaan [7, s. 7]. Kirjoittaja käyttää tästä englanninkielistä ilmaisua *hard navigation*, joka on hänen mukaansa perinteisen websovelluksen tapa navigoida sivustollaan. Kun ensimmäinen näkymä on luotu, yhden sivun sovellus mahdollistaa käyttäjän navigoida sivuston sisällä ilman uusia sivun latauksia palvelimelta. Kirjoittaja käyttää tästä ilmaisua *soft navigation* ja se tapahtuu hänen mukaansa renderöimällä uusia sivuja ensimmäisen latauksen yhteydessä ladatuista resursseista. [7, s. 7].

Pilvipohjaisena järjestelmänä käyttäjillä, jotka toimivat asiakas-palvelin-mallin asiakkaina, on pääsy websovellukseen eri laitteiden selaimien kautta. [7, s. 3]. Asiakas-palvelin-mallissa asiakas viittaa selaimeen, joka käyttää palvelimen tarjoamia palveluja [27]. Asiakas-palvelin-mallin toimiessa perusarkkitehtuurina järjestelmän arkkitehtuuri suunniteltiin kolmitasoiseksi niin, että edustaohjelma (frontend), taustaohjelma (backend) ja tietovarasto muodostavat omat itsenäiset moduulinsa. [7, s. 5]. Kolmikerrosarkkitehtuuri on monikerrosarkkitehtuureista yleisin [4, katso 7, s. 3].

Tietovarasto perustuu MYSQL- ja mongoDB-tietokantoihin [7, s. 5]. Edustaohjelman kehitystyön perustuessa SPA-malliin sen arkkitehtuuriksi valittiin *MVVM* (*Model-View-ViewModel*) AngularJS-sovelluskehyksen kautta toteutettuna. [7, s. 5]. Arkkitehtuuri on esitetty kuvassa 2.




***Kuva 2.*** *Tutkimuksessa kehitetyn sovelluksen korkean tason arkkitehtuuri mukaillen [7].*

AngularJS-sovelluskehykselle suunniteltiin oma erillinen arkkitehtuurinsa, jossa käyttöliittymä, datan määritelmä ja logiikka erotetaan omiksi kokonaisuuksiksi. Tällainen arkkitehtuuri mahdollistaa tärkeiden osakokonaisuuksien erottamisen toisistaan niin, että

koodi on hyvin organisoitua ja ylläpidettävää. Seuraavat komponentit kuuluvat keskeisesti kyseiseen arkkitehtuuriin [7, s. 6]:

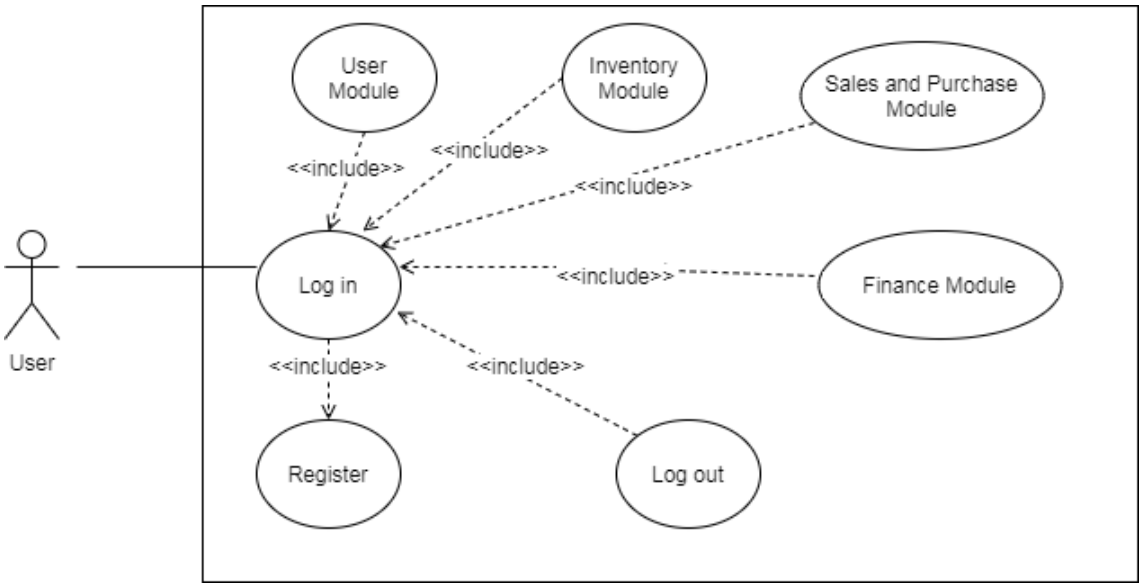
• View (Näkymä) —vastaa käyttöliittymästä näyttämällä HTML-sivun sisällön yhdessä CSS-tiedostoissa määriteltyjen tyylien kera. Se vastaa myös käyttäjän syötteiden vastaanottamisesta ja ylipäänsä vuorovaikutuksesta tämän kanssa ja siihen perustuen datan muutoksista Malliin molempiin suuntiin datan päivityksen mahdollistaman two-way data binding-tekniikan avulla tai kontrollerifunktiota kutsumalla. Kyseinen tekniikka mahdollistaa tapahtumien kuuntelun ja samanaikaisen datan päivityksen komponenttien välillä [2].

• Controller (Kontrolleri)—sisältää sovelluslogiikan JavaScript-muodossa ja joka mahdollistaa datan muokkauksen Mallin kautta tai vaihtoehtoisesti tarvittavan toiminnon suorittamisen.

• Service (Palvelu)—sisältää vastaavat funktiot kuin kontrollerikin, paitsi että palvelun tärkein funktio vastaa vuorovaikutuksesta palvelimen kanssa ja CRUD (*Create, Read, Update and Delete*) - operaatioiden suorittamisesta Mallin datalle. Kyseisillä operaatioilla viitataan tyypillisesti tietokannoissa käytettäviin perusfunktioihin.

• Model—(Malli) edustaa dataa, jota voidaan käyttää Näkymien päivittämiseen ja datan lähettämiseen Kontrollerille *two-way data binding*-tekniikan avulla.

Aiemmin mainitut pieniin ja keskisuuriin yrityksiin luokiteltavien vähittäis- ja tukkukauppojen kannalta olennaiset perusominaisuudet suunniteltiin toteutettavan moduuleina. Loppujen lopuksi neljäksi päämoduuliksi suunniteltiin User-, Inventory-, Sales and Purchase- ja Finance-nimiset moduulit, jotka on esitetty kuvan 3 käyttötapauskaaviossa. [7, s. 6].




***Kuva 3.*** *Tutkimuksessa kehitetyn SPA-sovelluksen käyttötapauskaavio mukaillen [7].*

User-moduuli keskittyy käyttäjäinhallintaan ja järjestelmän käyttöoikeuslupiin, kun taas Inventory-moduuli palvelee varastonhallintaa hallinnoimalla varastosaldoja sekä kuhunkin tuotteeseen liittyviä yksityiskohtaisia tietoja, Sales and Purchase-moduulien mahdollistaessa järjestelmän käyttäjille tallentaa ja hallinnoida kaikkia myyntiin ja ostoihin liittyvää dataa [7, s. 6]. Finance-moduuli taas mahdollistaa liiketoimintaan liittyvien raporttien generoinnin. Edellä kuvattujen perusmoduulien lisäksi järjestelmän tietokantojen käyttöä varten on oma CRUD-moduulinsa. [7, s. 7].

Toteutuksessa käytetään siis AngularJS-sovelluskehystä työkaluna yhden sivun sovelluksen luomiseksi. Mitä tulee sen rakennesuunnitteluun, kutakin järjestelmän entiteeteistä on tarkoitus käsitellä erillisessä moduulissa, jotta toteutettava koodi olisi organisoidumpaa ja ylläpidettävämpää. Kukin moduuleista sisältää useita JavaScript- ja HTML-tiedostoja, joista kutakin käytetään tietyn entiteetin hallintaan. [7, s. 7].

Kirjoittaja selventää sovellusrakennetta seuraavasti [7, s. 8]:

• Service.js—sisältää koodin, joka toteuttaa REST-rajapinnat, jotka taas mahdollistavat molemminpuolisen yhteyden palvelimen suuntaan.

• Page.html—vastaa pääsivun käyttöliittymästä, joka taas vastaa kuhunkin entiteettiin kuuluvien oliolistojen näyttämisestä.

• Page Controller.js—vastaa pääsivulla tapahtuvasta käyttäjän ja järjestelmän väliseen vuorovaikutukseen liittyvästä logiikasta.

• Modal.html—vastaa käyttöliittymästä, joka mahdollistaa yksittäisen olion muokkauksen sitä varten luodun lomakkeen avulla.

• Modal Controller.js—vastaa logiikasta, joka liittyy Modal.html-sivulla tapahtuvaan käyttäjän vuorovaikutukseen järjestelmän kanssa.

Teknologiatestit todistivat, että kehitetyn sovelluksen SPA-versio tarvitsee monisivuista versiota vähemmän verkkopyyntöjä navigoidessa User-sivulta Role-sivulle. Tämä johtuu siitä, että SPA-sovelluksen ei tarvitse ladata useita resursseja, jotka on ladattu jo ensimmäisen sivun latauksen yhteydessä. Tulos osoittaa, että SPA-toteutus pystyy parantamaan suorituskykyä perinteiseen monisivuiseen sovellukseen verrattuna johtuen vähemmästä dataliikenteestä asiakas- ja palvelinpään välillä navigointitilanteissa. [7, s. 8].

Myös Gzip-pakkausta ja moduulien niputtamisessa käytettävän Webpack-tekniikan tehokkuutta testattiin. Suurimmat edut tiedostokokojen pienennyksissä havaittiin pienimpien tiedostojen tapauksissa, mutta latausajat pienenivät eniten suurten tiedostojen tapauksissa. Niputus Webpack-tekniikkaan perustuen vähensi verkkopyyntöjen määrää 54:stä yhteen, koska yksi tiedostonippu vastaa kaikkia yksittäisiä resurssitiedostoja. Tämä nopeuttaa latausaikaa. Niputusprosessi lisäksi vähentää datansiirron tarvetta tilanteissa, joissa niputetun tiedoston kokoa on merkittävästi pienennetty. [7, s. 9].

Hyväksymistestaus tehtiin pilottitestauksen ohella, joka taas tehtiin ajatellen pieniä ja keskisuuria yrityksiä kohdemarkkinana. Hyväksymistestauksen kriteereinä käytettiin muun muassa moduulien vaatimustenmukaisuutta. Sen tulosten yhteenvetona todettiin, että käyttäjät olivat tyytyväisiä sovellukseen kaikkien kriteerien osalta, vaikka kyse oli vasta prototyypistä, joka ei sisällä kaikkia lopulliseen sovellukseen suunniteltuja ominaisuuksia. [7, s. 10].

## MYO ET AL. (2020)

Tutkimuksessa [17] pyritään etsimään keinoa vähentää palvelinsovellusten kehittämisessä syntyviä työvoimakustannuksia. Nykyisin käytössä olevassa metodologiassa kehitystyö alkaa asentamalla kehittäjän tietokoneelle kaikki tarvittavat kehittämistyökalut, asentamalla lokaalipalvelin ja lataamalla kaikki tarvittavat tiedostot mukaan lukien lähdekoodi ja ohjelmakirjastot [6, 19, 24, katso 17, s. 1]. Sen jälkeen ohjelmoija kehittää tietyn osan palvelinsovelluksesta tietokoneelleen asennetussa kehittämisympäristössä, joka sekin vaatii usein asennus- ja konfigurointityötä ennen varsinaisen kehitystyön aloittamista. Kun osa palvelinsovelluksesta on valmis, kehittäjä lataa sen palvelimelle käyttäen versionhallintajärjestelmää ja suorittaa viimeiset ohjelmistotestit palvelimella. Jos tässä tilanteessa on tarve editoida palvelimelle ladattuja tiedostoja, kehittäjä voi käyttää kahta eri metodia; tiedostoa editoidaan kehittäjän tietokoneella ja ladataan uudelleen palvelimelle tai editoidaan tiedostoa suoraan palvelimelle konsolia käyttäen. Ensin mainitussa vaihtoehdossa kehittäjä joutuu lataamaan suuria määriä informaatiota monta kertaa

edestakaisin tietokoneensa ja palvelimen välillä. Toinen vaihtoehdoista ei ole kätevä. Lisäksi mainittua metodologiaa käyttäen sovelluksen virheenetsintä voi muodostua ongelmaksi – erityisesti koska kyseisessä tapauksessa on mahdotonta saada selville missä testauksen tai kokonpanon tuloksena löydetyt virheet sijaitsevat. Paikallisten kehittämisympäristöjen käytön yhteydessä voi esiintyä myös niiden lisäosiin liittyviin vaatimuksiin liittyviä ristiriitoja ja ajoittaisista päivitysten puutteista johtuvaa toimintojen heikentymistä. Lisäksi, jos kehittäjä käyttää työhönsä useampaa tietokonetta, niin kaikki yllä mainitut toimet täytyy tehdä useammin kuin kerran – erityisesti jos työ tehdään useilla erilaisilla alustoilla. Esimerkiksi työpaikan kehittämisympäristön siirtäminen etätyössä käytettävälle Linux-tietokoneelle vaatii koko asennuksen tekemistä alusta asti. [17, s. 1, 3].

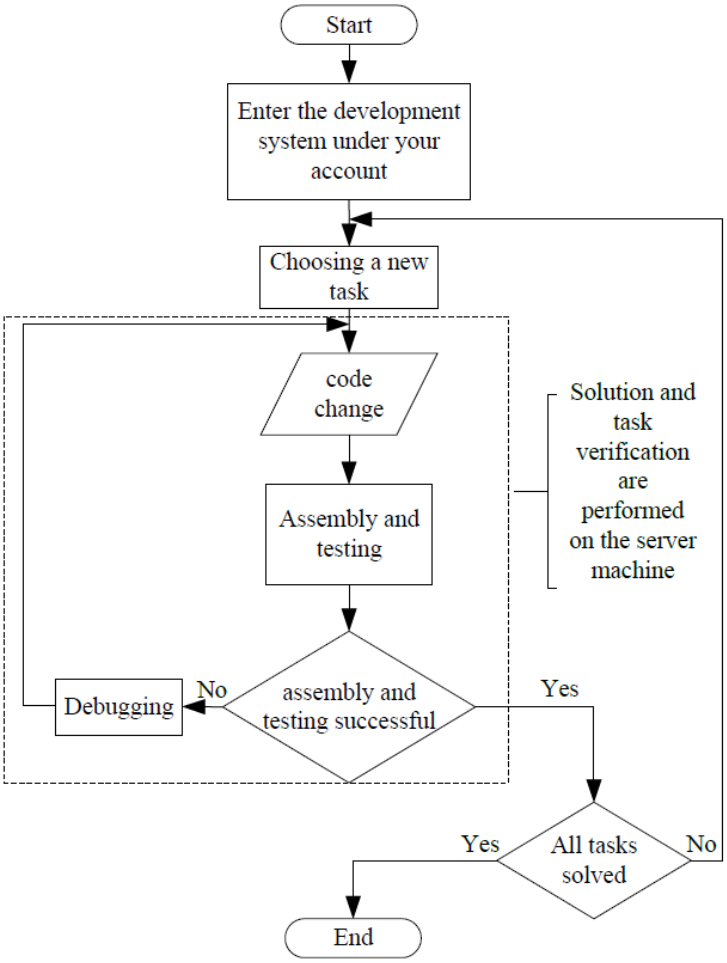
Kuvatussa metodologiassa merkittävä osa kehitystyöhön käytettävästä ajasta käytetään siis lähdekoodin lataamiseen palvelimelle – toisin sanoen, muuhun kuin itse kehitystyöhön kuluu merkittävästi resursseja [5, 15, katso 17, s. 1]. Nykyisin tosin yhä useammat sovellukset toimivat pilvessä kuten esimerkiksi ohjelmointiympäristöt (*IDE Integrated Development Environment*). Verkossa toimivat ohjelmointiympäristöt perustuvat selain-palvelin-rakenteeseen ja mahdollistavat kehittäjän kirjoittaa ohjelmia selaimen kautta. Ne voivat siten tuoda kätevyyttä kehittämisprosessiin, mutta niihin liittyy joitain haasteita; niistä puuttuu paljon sellaisia ominaisuuksia, joita esimerkiksi Eclipsen työpöytäversiossa on [16, 23, katso 17, s. 2]. Näistä tutkimuksessa mainitaan esimerkkinä uudelleen nimeäminen ja koodin automaattinen täydennys. Lisäksi verkossa toimiviin ohjelmointiympäristöihin liittyy tietoturvaan liittyviä ongelmia, joita ei työpöytäversioissa ole. [17, s. 1-2].

Edellä mainituista syistä on tutkimuksen mukaan tärkeää lisätä etäpalvelimelle tehtävän websovellusten kehitysnopeutta, paikallisille tietokoneille luotavien kehitysympäristöjen pystytysnopeutta, paikallisten kehitysympäristöjen ja etäpalvelimen välillä tapahtuvan tiedostojen synkronoinnin nopeutta sekä luoda metodologia, johon perustuen palvelinsovellusta voidaan kehittää suoraan palvelimelle [17, s. 2]. Ratkaistavista ongelmista merkittävin tässä ratkaisussa on tutkimuksen mukaan se, että tiedostojen editointi suoraan etäpalvelimelle ei ole yhtä tehokasta kuin perinteisessä ratkaisussa, jossa tiedostoja muokataan kehittäjän tietokoneella [17, s. 14].

Tutkimuksen mukaan on mahdollista, että lähes koko paikallisen kehittämisympäristön pystyttämiseen tarvittava työaika on poistettavissa, jos kunkin kehittäjän luoma lähde-

koodi ja tarvittavat ohjelmakirjastot sijaitsevat palvelimella ja kehittäjien tietokoneet toimivat ainoastaan projektin rajapintana. Tässä metodologiassa käyttäjän ei tarvitse huolehtia kehitystyössä käytettävistä työkaluista tai kirjastoista, vaan ainoastaan kehittäjien rajapintana projektiin toimivat paikallistietokoneet selainyhteyksineen täytyy konfiguroida. [17, s. 3].

Tämä uusi metodologia, jossa käytetään kehittäjän tietokonetta vain rajapintana projektiin, poistuu myös tarve ladata tiedostoja palvelimelle. Selaimen kautta tapahtuva tiedostojen muokkaus poistaa lisäksi etäpalvelimen konsolin käyttöön liittyvät ongelmat, kun selainrajapinnassa voidaan käyttää verkossa toimivia koodieditoreja. Suoraan palvelimelle kehitettävän koodin virheenetsinnässä voidaan tutkimuksen mukaan hyödyntää verkkopohjaisia ohjelmointiympäristöjä. Uusi metodologia on esitetty kuvassa 4. [17, s. 4].




***Kuva 4.*** *Tutkimuksen ehdottama uusi websovellusten kehitysmetodologia [17].*

Haitoiksi mainitaan, että kehittäjillä ei ole mahdollisuutta valita parhaiksi katsomiaan kehittämistyökaluja, mutta kyseinen haitta voidaan tutkimuksen mukaan minimoida sallimalla kehittäjille konfiguroida selainpäätä, jota he käyttävät kehittämiseen. Haitta ei lisäksi tutkimuksen mukaan olisi merkittävä monissa yrityksissä, joissa kyseinen konfigurointi on muutenkin standardoitu. [17, s. 3].

Tutkimuksessa vertaillaan vakiintunutta tapaa kehittää websovellusta ehdotettavaan metodologiaan määrittämällä edellä kuvatuille työvaiheille laskennalliset ajat ja approksimoimalla kehitystyöhön kuluva aika molemmilla menetelmillä. Tulosten mukaan kehitystyö on uudella metodologialla 1,61-2,89-kertaa nopeampaa verrattuna ehdotettavaan metodologiaan riippuen siitä, kuinka paljon tiedostojen vaihtoja palvelimelle käytännössä tarvitaan. Keskimääräistä tiedostojen vaihtoon määriteltyä arvoa käyttäen kehitystyön nopeus uudella metodologialla on yli kaksinkertainen. [17, s. 4-5].

## KARTTUNEN (2017)

Diplomityössä [13] tutkitaan, kuinka erilaiset prototypointitavat tukevat nopeatempoista tuotekehitystä ketterässä ohjelmistokehitysprojektissa. Siinä tutkitaan sekä prototypoinnin mahdollisuuksia itse ketterässä ohjelmistokehityksessä, että vaatimusmäärittelyssä. Tutkimus koostuu kahdesta osasta, joista ensimmäinen on kirjallisuusselvitys ja toinen koostuu haastatteluista ja toimintatutkimuksesta. [13, s. 3].

Tutkimuksen empiirinen osa toteutettiin erään yrityksen ohjelmistokehitysprojektissa, jossa tutkimuksen tekijä työskenteli. Haastatteluiden kohteina oli kyseisen projektin työntekijät ja ne toteutettiin puolistrukturoituna kyselynä siten, että haastateltavan annettiin ensin selittää omin sanoin, kuinka prototypointia hyödynnetään ketterissä ohjelmistoprojekteissa. Siihen perustuen, haastateltavaa varten muotoiltiin ja kysyttiin lisäkysymyksiä keskittyen haastateltavan ydinosaamiseen. [13, s. 33-34].

Tutkimustulosten mukaan ”prototypointia voidaan kehittää nopeatempoisten ketterien ohjelmistoprojektien tarpeisiin käyttämällä yksinkertaistettuja prototyyppejä sekä pienempiä ja fokusoituneempia prototyyppejä käyttöliittymäelementtien designin iteroinnin nopeuttamiseksi.” Lisäksi matalan tarkkuuden prototyypit ja osallistava suunnittelu nähtiin tuloksissa mahdollisiksi hyödyiksi ketterissä projekteissa. Isojen korkean tarkkuuden prototyyppien iteroinnin nopeuttaminen vaatii tutkimustulosten mukaan uusien työkalujen kehittämistä. Keskeisiin tutkimustuloksiin kuuluu myös, että ”prototypointi voi tukea ketterää vaatimusmäärittelyä esim. toimimalla dokumentaationa, helpottamalla kommunikaatiota ja tekemällä ns. ison kuvan selvemmäksi.” [13, s. 3].

Myös tutkimuksessa suoritetut haastattelut toivat lisäymmärrystä prototypoinnista ohjelmistokehityksessä. Haastatteluiden perusteella prototypoinnin tärkein tarkoitus on päästä kokeilemaan nopeasti ehdotetun idean toimivuutta ennen kuin siihen perustuen luodaan varsinainen tuote tai ominaisuus, joka taas vaatii enemmän aikaa ja kustannuksia kuin prototypointi. Useat haastateltavat vastasivat myös, että protypoinnin avulla voidaan kokeilla monia eri toteutusvaihtoehtoja ja arvioida mikä niistä toimii parhaiten. Lisäksi protypoinnilla voi kokeilla toimiiko jokin idea ylipäänsä. Eräs haastateltava vastasi myös, että prototypointiin tulisi suhtautua niin, että luodaan mahdollisimman pienellä vaivalla ja kustannuksilla prototyyppejä niin, että ne ovat tarvittaessa hylättävissä nopeasti ilman tunnesidettä. Haastateltavien käsityksiin kuului myös, että prototyypin tulisi keskittyä vain tärkeimpiin ominaisuuksiin ja käyttötapoihin; tämä on heidän mukaansa kriittistä varsinkin kehitysprosessin alussa, jolloin on tärkeää hahmottaa tuotteeseen todella tarvittavien ominaisuuksien minimimäärä. Vastauksien perusteella prototypoinnilla voidaan arvioida myös projektin budjettia ennen kuin on sitouduttu varsinaisen tuotteen kehittämiseen. [13, s. 39].

## 2.4 Yhteenveto, arviointi ja kehittämismetodologian muodostaminen

Kirjallisuuskatsauksen tuloksien perusteella pienten ja keskisuurten yritysten itse kehittämistä toiminnanohjausjärjestelmistä löytyy varsin vähän ennakkotapauksia ja ainoa konkreettinen hakutulos liittyy SPA-muotoiseen toteutukseen. Sen perusteella myös pienille ja keskisuurille yrityksille suunniteltavissa toiminnanohjausjärjestelmissä on huomioitava modulaarisuus perusrakenteena. Kyseisessä tutkimuksessa [7] toteutetussa prototyypissä muun muassa varastonhallinta on toteutettu omana moduulinaan. Tämä sopisi toimeksiantajan vaatimukseen, jonka mukaan kehitettävään toiminnanohjausjärjestelmään on oltava myöhemmin integroitavissa muun muassa varastonhallinta ja laskutus. Tutkimus ei kuitenkaan kuvaa moduulien yksityiskohtia, mutta kertoo niihin liittyvien hyväksymistestausten perusteella, että toteutetut moduuliratkaisut saivat hyvän vastaanoton kohdemarkkinoilla.

Vaikka tutkimuksessa toteutettu prototyyppi ei edusta samaa toimialaa kuin tämän työn toimeksiantaja, SPA-mallia päätettiin hyödyntää kehitettävän työnhallintajärjestelmän perustana. SPA-toteutuksesta on mainitun ennakkotapauksen mukaan hyötyä sovelluksen suorituskyvyn kannalta, kun sivustolla navigointiin ei liity muuta vuorovaikutusta palvelimen kanssa kuin yksittäisten sivujen sisällön lataus. Tästä on oletettavasti hyötyä myös ajatellen toimeksiantajan sovellukselle asettamaa vaatimusta toimia mobiililaitteilla. Gzip- ja Webpack-tekniikat pääteltiin suhteellisen hyödyttömiksi kehitettävän sovelluksen suorituskyvyn kannalta, sillä selainpäätä suunniteltiin kehitettävän yhteen

html- ja yhteen JavaScript-tiedostoon. Näin ollen ladattavien moduulien määrä jää pieneksi. Tässä työssä kehitettävässä SPA-toteutuksessa ei joka tapauksessa käytetä mainittuja tekniikoita.

Tutkimus tarjoaa mallin myös AngularJS-sovelluskehyksen käytöstä SPA-malliin perustuvan toteutuksen kehittämiseksi, mutta tässä työssä kehitettävä SPA-sovellus päätettiin toteuttaa ilman JavaScript-sovelluskehyksiä. Lisäksi tutkimuksen sovelluksessa noudatettua MVVM-arkkitehtuuria ei päätetty soveltaa, vaan kehitystyön pääperiaatteeksi valittiin toimintojen kehittäminen yksi kerrallaan alustavasti vain yhteen asiakaspään tiedostoon ja yhteen selainpään JavaScript-tiedostoon niin, että koko selainpään sovelluslogiikka toteutetaan funktioittain *jQuery*-kirjaston sisältämän *ready*-funktion sisään. Kyseisen funktion käyttötarkoitus on varmistaa, ettei toteutettava sovelluslogiikka yritä suorittaa toimintoja ennen kuin sivusto on siihen valmis. Tutkimuksen prototyypissä käytetty palvelinteknologia Node.js vastaa metodologian kehittämiseen liittyvää alustavaa ideaa käyttää diplomityöntekijän käyttämän Net9 Oy:n tarjoamaa Node.js-tuella varustettua Small-webhotellia kehitystyössä. Koska Node.js on alusta JavaScript-koodin tulkkaukseen ja sovellusten ajamista varten, myös palvelinpuolen tiedostot ovat JavaScript-tyyppisiä. Sen tarkempaa suunnitelmaa arkkitehtuurista ei tehty projektin alussa.

Etänä toteutettavista websovellusprojekteista löytyi vain vähän tietoa, mutta kirjallisuuskatsauksen ainoaksi lopulliseksi valinnaksi kyseiseen aiheeseen liittyen löydetty tutkimus [17] tarjoaa varteenotettavan vaihtoehdon etätoteutuksen perustaksi. Kyseisessä tutkimuksessa kehitetty metodologia liittyy tosin etäpalvelimiin sinänsä – ei webhotellipalveluihin, joihin usein sisältyy palveluntarjoajan määrittelemät websovellusten etähallintaan tarkoitetut ohjelmistot. Näin on myös tässä työssä käytettävän webhotellipalvelun osalta. Työssä kehitettävää websovellusta päätettiin näet lopulta ehdottaa kehitettävän suoraan diplomityöntekijän webhotelliin. Se sovittiinkin kehitettävän järjestelmän toteutustavaksi.

Tutkimuksessa kuvattuihin verkossa toimiviin koodieditoreihin ja ohjelmointiympäristöihin tutustuttiin vain pääpiirteittäin, jonka perusteella saatiin tietää, että ainakin osa verkossa toimivista ohjelmointiympäristöistä tukee SSH-yhteyttä. Käytettävissä ollut Small-webhotelli – toisin kuin sitä kalliimmat Net9 Oy:n tarjoamat webhotellituotteet ei sitä kuitenkaan tue. Koodin editointia päätettiinkin toteuttaa webhotellin *cPanel*-hallintapaneelin sisältämää tiedostonhallintassovellus *File Manager*ia käyttäen. Se sisältää perustoiminnot koodin editointia varten. Small-webhotellin cPanel sisältää myös sovelluksen *Setup Node.js App*, joka mahdollistaa Node.js-palvelinsovelluksen luomisen ja muun muassa ympäristön paketinhallintaan käyttämän npm-sovelluksen konfiguraatiomuutoksiin liittyvän komennon ***npm install*** suorittamisen [27]. Kyseinen komento päivittää esimerkiksi

lisätyt Node.js-moduulit, jotka määritellään tiedostoon *package.json*. Työssä käytettävä webhotelli tukee yhden MYSQL-tietokannan käyttöä. Tämän oletettiin riittävän kehitettävälle järjestelmälle.

Palvelinpään sovelluksen virheenetsinnässä päätettiin hyödyntää webhotellin cPanel-hallintapaneelin tarjoamaa mahdollisuutta tarkkailla tietokantojen sisältöä sovellusten toimintoja kokeiltaessa, tulostamalla palvelinsovelluksen palauttamaa dataa selaimen konsoliin ja näytölle sekä tarvittaessa diplomityöntekijän tietokoneelle luotavan lokaalipalvelimen avulla toteutettavan konsoliin tulostuksen avulla. Konsoliin ja näytölle tulostusta päätettiin hyödyntää myös selainsovelluksen virheenetsinnässä. Tärkein hyöty kehittää websovellusta suoraan webhotelliin on projektin etätoteutus; toimeksiantaja pääsee itse kokeilemaan ja tarkastelemaan kehitettyjä toimintoja sekä ominaisuuksia aidossa toimintaympäristössä riippumatta tämän omasta sijainnista.

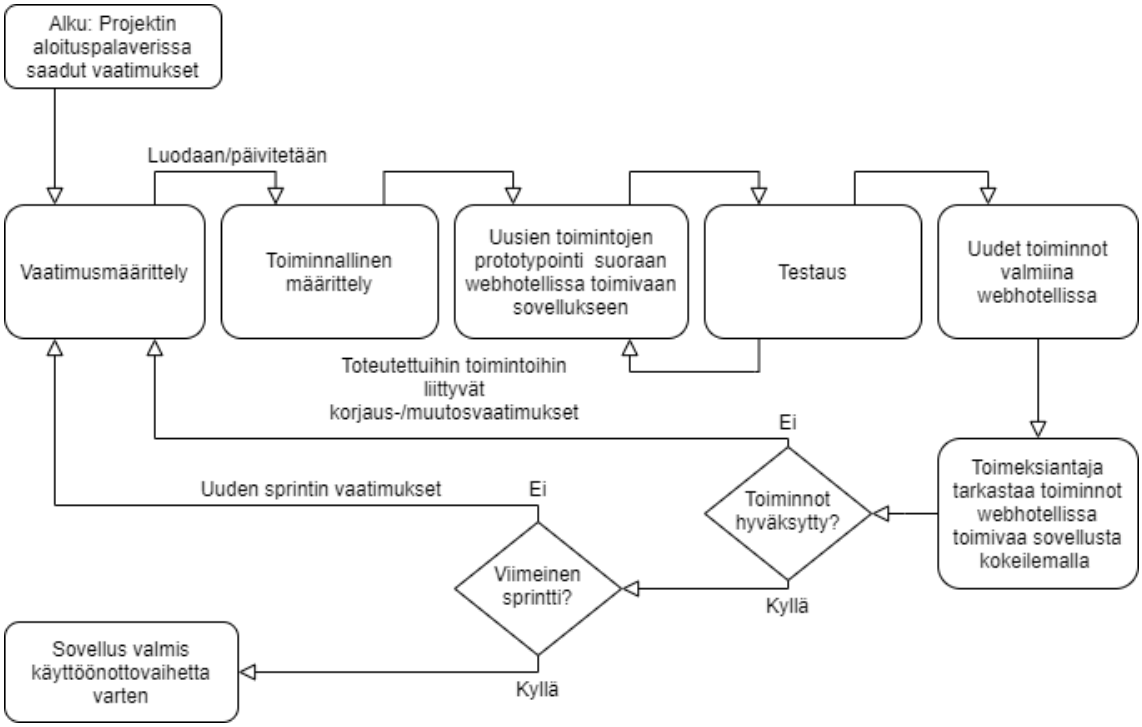
Prototypointiin liittyvä tutkimus [13] vahvisti, että eri toteutusvaihtoehtoja todella on hyödyllistä kokeilla nopeasti ilman suurta suunnittelupanosta. Prototypointia päätettiin hyödyntää niin, että kutakin tavoiteltua toimintoa kehitetään kokeilemalla mutta kuitenkin niin, että ensimmäinen toimiva versio on lähtökohtainen valinta tuotantoversiota varten eli valintaa ei tehdä useita eri prototyyppejä vertaamalla, ellei tilanne sitä vaadi.

Itsenäisen kehittäjän menetelmiin keskittyvän tutkimuksen [14] yhteys tähän työhön on yhdistelmä sen esittelemistä kolmesta itsenäisen kehittäjän menetelmästä. Metodologiaan päätettiin sisällyttää vastaavat yhteydenpitotavat kuin PXP:ssä, eli toimeksiantajaan ollaan yhteydessä sähköpostitse ja puhelimitse. SIP-iteraatio on muutoin vastaava kuin tässä työssä, paitsi että muutokset tallennetaan suoraan webhotellissa toimivaan kulloinkin kehitteillä olevaan versioon.

Agile Solossa käytettävä visualisointi toteutuu kehitettävässä metodologiassa toimeksiantajalle webhotellissa toimivan kehitysversion kautta näytettävinä toimintoina. Kehitysversio toimii samalla myös mallinnustapana asiakkaan kanssa kommunikointia varten. Kyseisen menetelmän neljän viikon sprintit ja viikoittaiset esittelyt asiakkaan kanssa eroavat kehitetystä metodologiasta siten, että sprinteille ei aseteta etukäteen pituutta ja esittelyjä asiakkaan kanssa tehdään vain tarvittaessa kesken sprintin – muutoin ne tehdään sprinttien lopuissa. Scrum-tyyppinen kehitysjono kuitenkin sisällytettiin metodologiaan. Tehtävien priorisointia tehdään pääasiassa sprinttien lopuissa, mutta tarvittaessa myös muulloin. Koska testaus tapahtuu kehittäjän toimesta kuten SIP-iteraatiossa, myös siltä osin Agile Solo eroaa kehitetystä metodologiasta. Viimeksi mainitussa menetelmässä tehtävä iteraatioiden suunnittelun yhteydessä tehtävää aika-arviointia suunnitel-

tiin ensin tehtävän tehtäväkohtaisesti yhdistäen *GanttProject*-sovelluksen *Gantt*-kaavioiden käyttö metodologiaan, mutta lopullisessa versiossa projektin edetessä siitä luovuttiin ja päädyttiin käyttämään tehtäväjonojen hallintaan soveltuvaa *Asana*-sovellusta ilman tehtävien tai edes sprinttien aika-arviointia.

Testivetoista kehitystapaa päätettiin olla hyödyntämättä. Koodin arvioinnissa käytetään vain itsearviointia. Metodologiaksi muotoutui lopulta sprintteihin perustuva malli, jossa toimeksiantajalta saatuihin toimintoihin ja ominaisuuksiin liittyviä vaatimuksia toteutetaan valmiiksi ja tämä käy ilmoituksen perusteella kokeilemassa ja tarkastelemassa niitä. Toimeksiantajan arvioitua toteutumaa siihen joko tehdään muutoksia tai siirrytään seuraavaan sprinttiin vielä toteuttamattomat tai toimeksiantajan esittämät uudet vaatimukset lähtökohtana. Kehitetty metodologia SPA-malliin perustuvan työnhallintajärjestelmän kehittämiseksi on esitetty kuvassa 5. Toiminnallisen määrittelyn ajateltiin päivittyvän projektin edetessä, mutta sen dokumentoinnin suhteen ei tehty suunnitelmia.




***Kuva 5.*** *Kehitetty metodologia työnhallintajärjestelmän toteuttamiseksi etänä.*

# 3. ARVIOINTIKRITEERIT JA MENETELMÄT

Projektin arviointi kattaa tässä työssä paitsi projektinhallinta- ja sovelluskehitysprosessien, myös työtuloksien sekä käytettävien työkalujen arvioinnin. Prosessien arviointikriteerinä käytetään luvussa 3.1 esiteltävää pienorganisaatioille suunnattua standardia ja muut arviointikriteerit on kuvattu aliluvussa 3.2 kaikkien arviointimenetelmien ollessa koottu viimeiseen alilukuun.

## 3.1 Prosessien arviointikriteeri

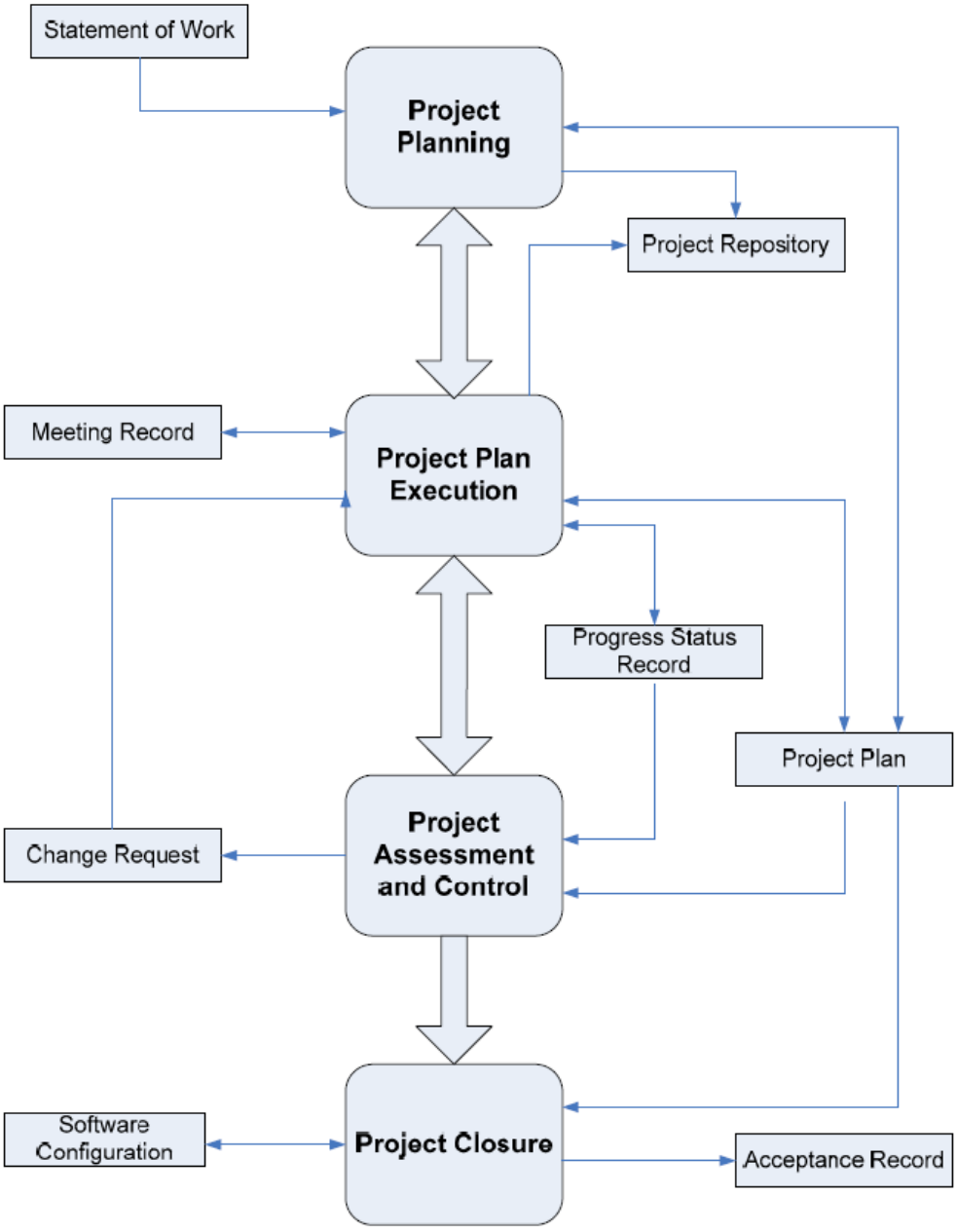
Standardin ISO/IEC 29110 kehittämisen taustalla on tutkimukset, joiden mukaan pienyritysten ja muiden vastaavan kokoluokan organisaatioiden (*Very Small Entities -VSE*) on vaikeaa hyödyntää olemassa olevia standardeja ohjelmistokehityksessään, sillä ne ovat suunniteltu suuremmille organisaatioille. Standardi määrittelee kyseisten organisaatioiden enimmäiskooksi 25 henkilöä. Se kehitettiin siten, että olemassa olevista standardeista koottiin pienorganisaatioiden tarpeisiin parhaiten sopivat osat; esimerkiksi prosesseja koskevia osia otettiin standardista ISO/IEC 12207. Standardin ISO/IEC 29110 tarkoitus on parantaa tuotteiden laatua ja prosessien tehokkuutta. Sen tarkoitus ei ole sulkea pois varsinaisia ohjelmistokehitysmenetelmiä tai prosesseja kuten vesiputousmallia tai ketterän kehityksen menetelmiä. [12, s. 6].

Standardi ISO/IEC 29110 koostuu viidestä osasta, joista ohjeistus ISO/IEC TR29110 Software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 5-1-1: Management and engineering guide: Generic profile group: Entry profile tarjoaa oppaan projektinhallintaan ja ohjelmistokehitysprosessiin [12, s. 9]. *TR* on lyhenne sanoista *Technical report*, jolla viitataan standardin yleiskatsauksiin ja ohjeistuksiin [12, s. 7]. *Entry profile* viittaa erääseen standardin profiileista ts. erilaisista versioista, joka on suunnattu esimerkiksi startup-yrityksille, jotka kehittävät yksittäistä sovellusta ilman erityistä riskiä projektissa, jonka suuruus on pienempi kuin 6 henkilötyökuukautta [12, s. 7].

Edellä mainittu ohjeistus sisältää ohjeet kahden ohjelmistoprojektin ehkä tärkeimmän prosessin; projektinhallintaprosessin (*project management process*) ja ohjelmiston toteutusprosessin (*software implementation process*) toteuttamiseksi. Ohjeistus tarjoaa systemaattisen tavan toteuttaa kyseiset prosessit. Siinä on määritelty muun muassa prosessien tavoitteet sekä niiden aktiviteetit syötteineen (*Input product*) ja aikaansaannoksineen (*Output product*). Se sisältää jopa tehtävälistan vastuineen molemmille prosessille. Ohjeistuksen hyödyntämisen lähtökohtana tulisi olla dokumentoitu työn kuvaus (*Statement of work*), joka saadaan asiakkaalta. [12, s. 9, 17, 26].

### 3.1.1 Projektinhallintaprosessi

Projektinhallintaprosessin tarkoitus on luoda systemaattisesti ohjelmiston toteutusprojektin tehtävät, jotka mahdollistavat projektin tavoitteiden saavuttamisen laadun, aikataulun ja kustannusten suhteen. Se koostuu neljästä aktiviteetista, jotka ovat **Projektisuunnittelu** (*Project planning*), **Suunnittelun toimeenpano** *(Project plan execution)*, **Projektin arviointi ja kontrollointi** *(Project assessment and control)* sekä **Projektin päättäminen** (*Project closure*). Kronologisessa järjestyksessä niistä ovat vain ensin ja viimeisenä mainitut aktiviteetit. Projektinhallintaprosessi on esitetty kaaviona kuvassa 6. [12, s. 12, 17].

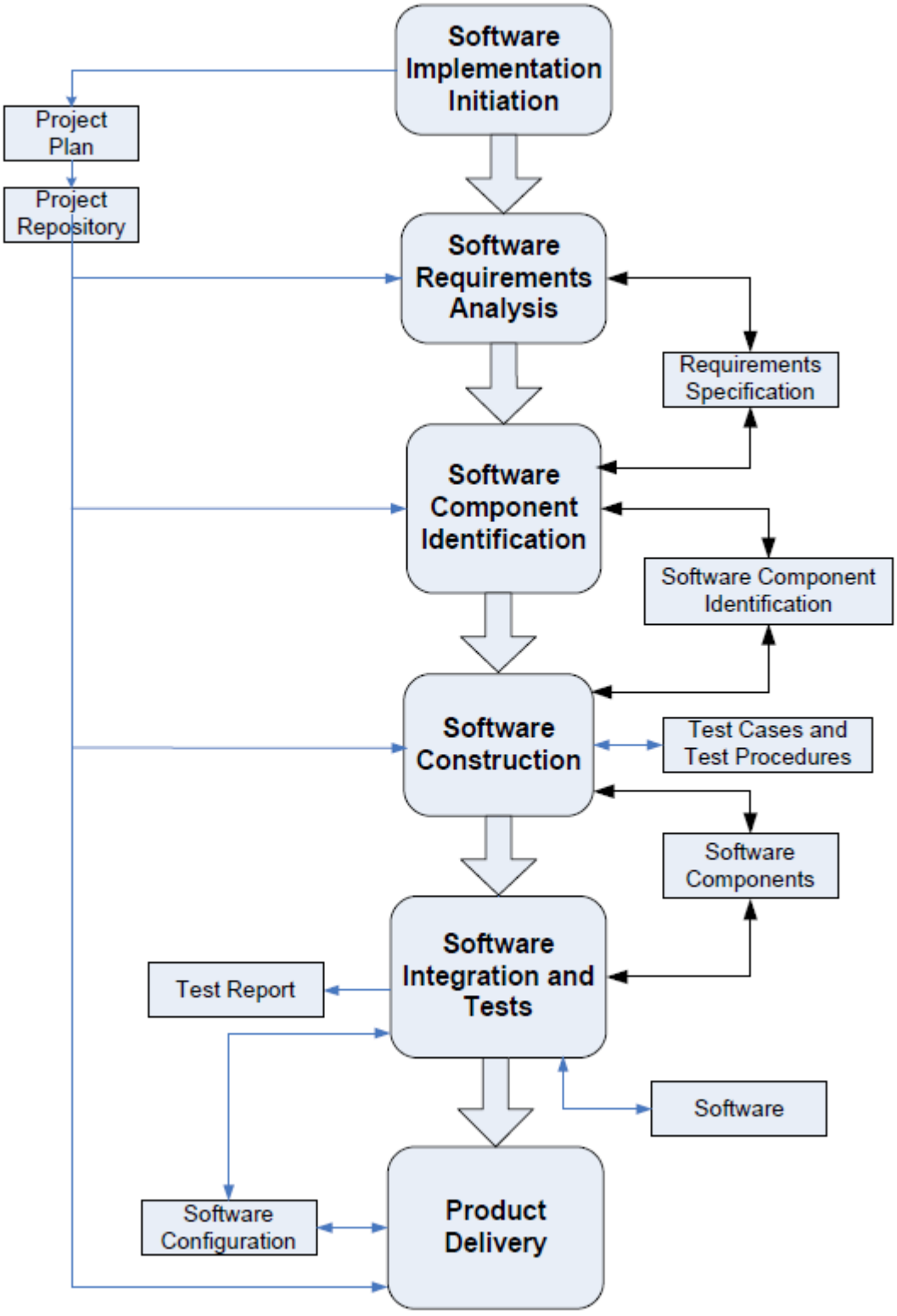



***Kuva 6.*** *Projektinhallintaprosessi [12, s. 17].*

Esimerkiksi **Projektisuunnittelu**-aktiviteetin toisessa tehtävässä (PM.1.2) tunnistetaan katselmoidusta työn kuvauksesta tarpeita, jotka liittyvät projektin tuotoksien ja niiden ohjelmistokomponenttien tuottamiseksi. Niiden perusteella luodaan tarkoituksenmukaiset tehtävät. Sen jälkeen tehtävät sisällytetään sovelluskehitysprosessiin asiakkaan ja kehitystiimin kanssa tehtävien tarkistus-, hyväksyntä- ja katselmointitehtävien kera [12, s. 18].

### 3.1.2 Sovelluskehitysprosessi

Sovelluskehitysprosessin tarkoitus on vaatimusanalyysin, ohjelmistokomponenttien tunnistamisen, ohjelmiston rakentamisen, integroinnin, testauksen ja tuotteen toimitukseen liittyvien toimenpiteiden suorittaminen kehitettävälle sovellukselle asetettujen vaatimusten mukaisesti. Se koostuu seitsemästä aktiviteetista, jotka ovat **Prosessin aloitus**, **Vaatimusten analyysi**, **Ohjelmistokomponenttien tunnistus**, **Ohjelmiston rakentaminen**, **Kokoonpano ja testaus** sekä **Tuotteen toimitus**. Sovelluskehitysprosessi on esitetty kaaviona kuvassa 7. [12, s. 22, 26].




***Kuva 7.*** *Sovelluskehitysprosessi [12, s. 26].*

Esimerkiksi **Vaatimusten analyysi** - aktiviteetin tehtävän SI.2.2 ydinkontribuutio on toiminnallisen määrittelyn dokumentointi tai vaihtoehtoisesti sen päivittäminen. Sen toteuttamiseksi standardi ohjeistaa tunnistamaan mahdollisia vaatimuslähteitä kuten asiakas, sovelluksen käyttäjät, aiemmat järjestelmät jne. Lisäksi se ohjeistaa kokoamaan ja analysoimaan vaatimukset niiden laajuuden ja toteutettavuuden määrittämiseksi. Standardissa on asetettu tehtävän suorittajiksi projektiryhmä ja asiakas. Tehtävän toimintoihin kuuluu myös kehitteillä olevan/aiemmin toteutetun toiminnallisen määrittelyn oikeellisuuden tarkistus työn kuvaukseen nähden sekä sen määrittelemien toimintojen testattavuus. Lopuksi kehitteillä oleva/aiemmin toteutettu toiminnallinen määrittely generoidaan/päivitetään lopulliseen muotoonsa. [12, s. 27].

**Ohjelmistokomponenttien tunnistus** taas on aktiviteetti, joka muuntaa ohjelmistovaatimukset ohjelmistokomponenttien arkkitehtuuriksi. Esimerkiksi tehtävässä SI.3.3 tilanteesta riippuen joko dokumentoidaan tai päivitetään aiemmin dokumentoidut (tunnistetut) ohjelmistokomponentit. Se tapahtuu analysoimalla toiminnallista määrittelyä ja sen pitäisi tuottaa paitsi dokumentoidut ohjelmistokomponentit, myös niiden järjestelyn mahdollisissa osajärjestelmissä sekä ohjelmistokomponenttien määrittämät ulkoiset ja sisäiset rajapinnat. Tehtävän lopussa paitsi ohjelmistokomponenttien, myös niiden rajapintojen yksityiskohdat pitäisi olla selvillä ohjelmiston rakentamiseksi. Tehtävä kuuluu kehitystiimin vastuulle. [12, s. 28].

## 3.2 Muiden tekijöiden huomiointi

Projektissa käytettäviä työkaluja tarkastellaan erikseen käyttämällä kriteerinä niiden käytettävyyttä kehitetyn metodologian yhteydessä. Työkaluja ovat ensinnäkin webhotellin hallinnointiin tarkoitettuun cPanel-sovellukseen kuuluva tiedostonhallintassovellus File Manager, joka sisältää myös perustoiminnot koodin editointia varten. Lisäksi tarkastellaan lokaalipalvelimen käytön soveltuvuutta osana kehitettyä metodologiaa. Mainituista työkaluista käytetään taulukossa lyhenteitä KE ja LP vastaavasti. Myös kehitettyä työnhallintajärjestelmää tarkastellaan. Koska kyseinen järjestelmä perustuu suurelta osin toimeksiantajan suunnitelmiin, sen toimintojen käytettävyyttä tai tarkoituksenmukaisuutta ei arvioida. Sen sijaan sovittujen toimintojen ja ominaisuuksien toteutuminen arvioidaan käyttöönottoa varten toteutetun ohjelmistotestauksen perusteella. Tästä käytetään taulukossa merkintää ERP1. Sen lisäksi sovelluksen tarkastelukriteereinä käytetään laajennettavuutta, koodin uudelleenkäytettävyyttä ja luettavuutta. Mainituista kriteereistä käytetään merkintöjä ERP2, ERP3 ja ERP4 vastaavasti. Laajennettavuus ja uudelleenkäytettävyys arvioidaan ensisijaisesti sovellukselle projektin alussa asetettujen laajennettavuusvaatimusten kannalta.

## 3.3 Arviointimenetelmät

Toteutettua projektia on tarkoitus verrata standardin ISO/IEC TR29110 osan 5-1-1 ohjeisiin niin, että projekti käydään läpi erikseen kyseisen ohjeistuksen näkökulmasta. Tämä tarkoittaa, että toteutettua projektia verrataan mainitun ohjeistuksen määrittelemiin prosesseihin. Se tapahtuu käytännössä niin, että prosessiaktiviteettien sisältämien tehtävien toteutumista arvioidaan. Arvioinnissa kuitenkin huomioidaan vain sellaiset tehtävät ja niiden osat, joita on tarkoituksenmukaista arvioida yhden henkilön projektissa. Esimerkiksi projektinhallintaprosessin tehtävää PM.1.5, jossa luodaan kehitystiimin kokoonpano ja asetetaan sille vastuut ja velvollisuudet, ei ole tässä tapauksessa tarkoituksenmukaista arvioida. Tämäntyyppiset tehtävät rajataan kokonaan arvioinnin ulkopuolelle.

Lisäksi esimerkiksi tehtävässä PM.1.2 (luku 3.1.1, s. 25) on huomioitava, että erillistä projektipäällikköä ja kehitystiimiä ei tässä projektissa ole, vaan koko projektista vastaa yksi henkilö. Tällöin erilliseen kehitystiimiin liittyvät kohdat tehtävästä rajataan arvioinnin ulkopuolelle, eli sen vaikutus arviointiin on neutraali. Vastaavaa arviointia käytetään muissakin tehtävissä, joissa oletetaan, että projektin käytössä on erillinen kehitystiimi. Aktiviteetteihin **Projektin päätös** ja **Tuotteen luovutus** liittyvät tehtävät rajataan kokonaan arvioinnin ulkopuolelle, sillä ne liittyvät pääosin tuotteen luovutukseen, jota ei toteutettu.

Arviointiasteikkona käytetään mm. standardin ISO/IEC 33020 käyttämää nelitasoista N-P-L-F-asteikkoa [11]. Kirjainlyhenteet on muodostettu seuraavista tasoista: Not achieved (0–15%), Partially achieved (>15–50%), Largely achieved (>50–85%) ja Fully achieved (>85–100%) [11]. Edellä mainituille arviointiin sopimattomille osille käytetään merkintään n/a (not applicable). Arvioinnin tuloksena aktiviteetit lisäksi järjestetään merkittävyysjärjestykseen niin, että aktiviteetin valintaan johtaneet tehtävät on mainittu aktiviteetin kohdalla. Aktiviteettien ja niiden valintaan johtaneiden tehtävien joukosta pyritään myös löytämään keskinäisiä syy-seuraus-suhteita.

Myös projektissa käytettyjen työkalujen ja kehitetyn järjestelmän tarkastelussa käytetään edellä kuvattua N-P-L-F-asteikkoa. Arvioinnin lopuksi kiteytetään projektin suurimmat heikkoudet koko arvioinnin perusteella ja pyritään etsimään myös niistä keskinäisiä yhteyksiä ja juurisyitä.

# 4. TYÖNHALLINTAJÄRJESTELMÄN TOTEUTUS

SPA-mallia noudattava työnhallintajärjestelmä toteutettiin sovittua metodologiaa käyttäen. Ensimmäiseen alilukuun on koottu projektin eteneminen kronologisesti niin, että projektin etenemistä kuvataan toimeksiantajan kanssa käydyn yhteydenpidon avulla kolmen viimeisimmän aliluvun keskittyessä itse toteutuksen kuvaukseen.

## 4.1 Projektin eteneminen ja yhteydenpito toimeksiantajaan

Tähän alilukuun on koottu toimeksiantajan kanssa toteutunut transaktio, joka toteutui suurimmalta osin sähköpostitse lukuun ottamatta joitain puheluita, joihin viitataan erikseen. Kaikkia puheluita ei kuitenkaan onnistuttu identifioimaan yhteydenpitoon käytetyn puhelimen historiatiedoista, sillä toimeksiantajalla oli lopulta käytössään useita liittymiä yhteydenpitoon, joista vain yhden numero tallennettiin diplomityöntekijän puhelimen yhteystietoihin. Puheluita kertyi joka tapauksessa alle kymmenen koko projektin aikana.

Alla kuvattu viestintähistoria sisältää merkittävän osan kehitystyön vaiheista sekä sen aikana nousseista kysymyksistä. Kutakin kuukautta käsittelevän osan alkuun on lisätty taulukot kuvaamaan sähköpostitse tapahtunutta transaktiota toimeksiantajan kanssa. Esimerkiksi taulukkoon 2 on kuvattu syyskuun 2019 transaktiohistoria.

### SYYSKUU 2019

***Taulukko 2. Transaktio toimeksiantajan kanssa syyskuun 2019 osalta.***

| | |
|---|---|
| 12.09.2019 | Hakemus toimeksiantajalle sähköpostitse |
| 13. – 28.09.2019 | Sähköpostiviestiketju toimeksiantajan kanssa |
| 30.09.2019 | 1. tapaaminen toimeksiantajan kanssa |

Projektin ensimmäinen transaktio tapahtui lähettämällä hakemus toimeksiantajalle. Myöhemmin asiaan palattiin myös puhelimitse (19.09.2019), mutta pääosa yhteydenpidosta tapahtui viestiketjussa, jossa tiedusteltiin yrityksen nykyisestä työnhallintajärjestelmästä sekä millaisen järjestelmän kehittämisestä on kyse. Vastausten perusteella mm. tutustuttiin käytössä olleeseen Tribuni-toiminnanohjausjärjestelmään, josta ei tosin enää ollut käytössä kuin asiakasrekisteri. Viestiketjussa käsiteltiin myös diplomityöprojektiin liittyviä aloitustoimia. Lisäksi tiedusteltiin, oliko järjestelmä tarkoitus integroida yrityksen jo käytössä olevaan websivustoon vai kokonaan erilliseen webhotelliin. Lopuksi todettiin, että kehitettyä järjestelmää kannattaa käyttää erillisessä webhotellissa.

Kasvotusten toteutetussa tapaamisessa toimeksiantaja kertoi aiemmasta kokemuksestaan kodinkoneiden huoltoliikkeille suunniteltuun Tribuni-toiminnanohjausjärjestelmään liittyen. Toimeksiantaja kertoi myös ideoistaan sen pohjalta. Vaatimuksista selvitettiinkin

merkittävä osa jo aloitustapaamisessa, jossa selvisi ensimmäisenä, että työnhallintajärjestelmän tulee sisältää valikkorivi, joiden valikkovalinnat ovat **Asiakasrekisteri**, **Luo toimenpide**, **Selaa toimenpiteet** ja **Kalenteri**. Kahteen ensiksi mainittuun valikkovalintaan tuli kuulua myös omat lomakkeensa, joiden sisällöstä tehtiin lyijykynäluonnokset. Aloituspalaverin ja myöhemmän vaatimusmäärittelyn perusteella määritellyt vaatimukset on esitetty liitteessä A. Nämä kattavat pääosan vaatimuksista ennen projektin viimeistä kasvotusten toimeksiantajan kanssa pidettyä tapaamista jossa toimeksiantaja esitti viimeiset vaatimuksensa kehitettävälle työnhallintajärjestelmälle. Kaikista vastaanotetuista vaatimuksista on johdettu toiminnallinen määrittely, joka on kuvattu liitteessä B.

Sovittujen toimintojen valmistumiselle asetettiin takarajaksi 30.4.2020, mutta toimeksiantaja ilmaisi projektin alun tapaamisessa, että koko työnhallintajärjestelmän ei välttämättä tarvitse olla valmis juuri projektin suunniteltuna valmistumisajankohtana, koska sillä on joka tapauksessa kiinnostusta sen jatkokehitykseen ohjelmiston perusversion toteutuksen jälkeenkin. Kyseisessä tapaamisessa näet keskusteltiin alustavasti perusversioon integroitavista tai kehitettävistä varastonhallintaan ja laskutukseen liittyvistä toiminnoista. Toimeksiantajan mukaan kyseiset toiminnot olisi joka tapauksessa tarpeen sisällyttää jossain vaiheessa tavalla tai toisella yrityksen käyttämään tietojärjestelmään. Lokakuun 2019 transaktio on koottu taulukkoon 3.

## LOKAKUU 2019

***Taulukko 3. Transaktio toimeksiantajan kanssa lokakuun 2019 osalta.***

| | |
|---|---|
| 01.10.2019 | Zip-paketin lähetys toimeksiantajalle |
| 02.10.2019 | Toimeksiantajan hyväksyntä |
| 03.10.2019 | Zip-paketin lähetys toimeksiantajalle |
| 05.10.2019 | Kuvankaappauksen lähetys edelliseen liittyen |
| 07.10.2019 | Yritys ilmoitti ongelmista zip-tiedostoon liittyen |
| 08.10.2019 | Viestiketju aiheesta BudgetSMS |
| 08.10.2019 | Toimeksiantajan hyväksyntä BudgetSMS - palvelulle |
| 09.10.2019 | Vahvistus BudgetSMS-suunnitelmiin liittyen |
| 11.10.2019 | Ehdotus toimeksiantajalle Tekstari.fi:n liittyen |
| 12.10.2019 | Yrityksen hyväksyntä Tekstari.fi-palvelulle |
| 13.10.2019 | Viestiketju liittyen Node.js-alustaan |
| 14.10.2019 | Viestiketju liittyen Tekstari.fi-käyttäjätiliin |
| 15.10.2019 | Ilmoitus onnistuneesta SMS-testistä |
| 15.10.2019 | Toimeksiantaja hyväksyi kehitystyössä lähetettävät tekstiviestit |
| 18.10.2019 | Viestiketju liittyen mm. autentikoinnin etenemiseen |
| 21.10.2019 | Ilmoitus lisätyön tarpeesta SMS-palveluun liittyen |
| 21.10.2019 | Toimeksiantaja hyväksyi SMS-palveluun liittyvän lisätyötarpeen |
| 22.10.2019 | Viestiketju mm. SMS-palvelun kokeiluihin sekä kehitysympäristön käyttöönottoon liittyen |
| 30.10.2019 | Viestiketju liittyen autentikointivaihtoehtojen kokeiluihin |
| 31.10.2019 | Ilmoitus mm. onnistuneesta Let's Encrypt-salauksen käyttöönotosta |

Lokakuussa 2019 aloitettiin ensimmäisten toimintojen toteutus, joita visualisoitiin lähettämällä toimeksiantajalle tarvittavat käyttöliittymän tiedostot zip-paketteina. Sen käytössä ilmeni kuitenkin ongelmia, sillä käyttöliittymän kaikki ominaisuudet eivät vastanneet kehittäjän tietokoneella kokeiltua zip-tiedoston avulla kokeiltua visualisointia. Myöhemmin paljastui, että toimeksiantaja ei ollut purkanut zip-pakettia, mutta visualisointitavasta ei keskusteltu enempää. Sen jälkeen toimeksiantajalle ehdotettiin BudgetSMS-tekstiviestipalvelun käyttöä, jonka tämä hyväksyi 8.10.2019. Seuraavana päivänä lähetettiin vielä vahvistusviesti toimeksiantajalle. Sitten diplomityönohjaajalta saatiin vihje, jonka mukaan kotimaisen tekstiviestipalvelun käyttö olisi tietoturvasyistä suositeltavaa. Toimeksiantaja hyväksyikin kyseisen vihjeen perusteella ehdotetun Tekstari.fi-palvelun käytön kehitettävässä järjestelmässä.

Sen jälkeen siirryttiin Node.js-kehittämisympäristön käyttöönottoon, johon liittyen suunniteltiin jopa lopullisen Node.js-tuotantoympäristön luomista toimeksiantajalle, mutta kehitystyötä päätettiin lopulta jatkaa diplomityöntekijän webhotellin vastaavassa ympäristössä. Myöhemmin toimeksiantajaa mm. muistutettiin käyttäjätilin avaamisesta Tekstari.fi-palveluun, ilmoitettiin Tekstari.fi-palvelun rajapintaa muistuttavalla BudgetSMS:n rajapinnalla tehtyjen kokeilujen onnistumisesta sekä autentikointiin liittyvistä kehitysvaiheista. Toimeksiantaja mm. hyväksyi tekstiviestipalvelulla tehtävät lisätestit, vaikka se aiheuttaakin kustannuksia tälle. Sitten toimeksiantajaa informoitiin Tekstari.fi-palvelun vaatimasta ja kyseisen palveluntarjoajan toimittamasta erillisestä koodista rajapinnan käyttöä varten, jonka vuoksi tekstiviestipalvelun käyttöönotto vaatii odotettua enemmän aikaa. Myös onnistuneesta Tekstari.fi-palveluun liittyvästä kokeilusta ja sen käytön vaatimista muutoksista sovellukseen sekä kehitysvaiheiden visualisoinnin jatkumisesta diplomityöntekijän webhotellia hyödyntäen ilmoitettiin. 22.10.2019 aloitetussa viestiketjussa toimeksiantaja antoi myös lisätoiveita teknisiin ratkaisuihin liittyen. Kuun lopussa toimeksiantajaa informoitiin vielä mm. eri autentikointivaihtoehtojen kokeilujen etenemisestä, onnistuneesta Let's Encrypt-salauksen käyttöönotosta kehitysympäristössä sekä Basic-autentikointitavan valinnasta. Marraskuun 2019 transaktio on koottu taulukkoon 4.

## MARRASKUU 2019

***Taulukko 4. Transaktio toimeksiantajan kanssa marraskuun 2019 osalta.***

| | |
|---|---|
| 01.11.2019 | Ilmoitus onnistuneesta websovelluksen autentikoinnin onnistuneesta käyttöönotosta |
| 01.11.2019 | Toimeksiantaja hyväksyi autentikointitoiminnallisuuden |
| 06.11.2019 | Ilmoitus webhotellin uudesta sivustojärjestelystä |
| 09.11.2019 | Ilmoitus liittyen *Scheduler*-kalenterimoduulin alustavaan valintaan |
| 09.11.2019 | Toimeksiantaja hyväksyi *Scheduler*-kalenterimoduulin valinnan |
| 12.11.2019 | Ilmoitus kalenterin perustoiminnallisuuksien toimivuudesta webhotellissa |
| 12.11.2019 | Toimeksiantaja hyväksyi perustoiminnallisuudet |

Kuun alussa toimeksiantajaa informoitiin autentikoinnin onnistuneesta käyttöönotosta, jonka tämä hyväksyi kokeilunsa perusteella. Myöhemmin viestittelyä jatkettiin informoimalla yritystä webhotellin uudesta sivustojärjestelystä ja **Asetukset**- ja **Luo toimenpide**-valikkovalintoihin liittyvien toimintojen ja ominaisuuksien kehittämisen etenemisestä. Tämän jälkeen siirryttiin kalenteritoteutuksen suunnitteluun ja kokeiluihin, joiden perusteella toimeksiantajalle kerrottiin *dhtmlx*-kirjaston tarjoaman *Scheduler*-kalenterimoduulin alustavasta valinnasta ja sillä tehdyistä onnistuneista kokeiluista. Niiden ja toimeksiantajan vaatimusten perusteella ehdotettiin kyseiseen moduuliin liittyvää teknistä ratkaisua, jonka toimeksiantaja hyväksyi. Kuun viimeisessä yhteydenpidossa ilmoitettiin kalenterin perustoiminnallisuuksien toimivuudesta webhotellissa, saatiin niille hyväksyntä sekä sovittiin toimeksiantajan ehdotuksen perusteella, että seuraava tapaaminen pidettäisiin vasta, kun kaikki siihen mennessä sovitut toiminnallisuudet ovat valmiit. Joulukuun 2019 ja tammikuun 2020 transaktio on koottu taulukkoon 5.

## JOULUKUU 2019 - TAMMIKUU 2020

***Taulukko 5. Transaktio toimeksiantajan kanssa joulukuusta 2019 tammikuuhun 2020.***

| | |
|---|---|
| 03.-06.12.2019 | Viestiketju liittyen Luo toimenpide-valikkovalintaan liittyen |
| 17.12.2019 | Varmistus seuraavan tapaamisen ajoittumisesta seuraavan vuoden puolelle |
| 17.12.2019 | Toimeksiantaja vastasi edelliseen viestiin |
| 13.01.2020 | Ilmoitus Luo toimenpide-valikkovalinnan toimintojen valmistumisesta |
| 13.01.2020 | Toimeksiantaja vastasi edelliseen viestiin |
| 21.01.2020 | Kysymys aiempaan toiminnanohjausjärjestelmään liittyen |
| 21.01.2020 | Toimeksiantaja vastasi edelliseen viestiin |
| 31.01.2020 | Kysymys toimeksiantajalle kalenteritoteutuksen yksityiskohtiin liittyen |
| 31.01.2020 | Toimeksiantaja vastasi edelliseen viestiin |

Luo toimenpide-valikkovalintaan liittyvässä kehitystyössä esiin noussut tarve uudelle toimenpiteelle lisättävästä ajankohdasta johti viestiketjuun, jonka tuloksena sovittiin, että alustavaksi toimenpiteen aloitusajankohdaksi generoidaan tallennushetken aika ja lopetusajankohdaksi tunti lisää aloitusajankohtaan nähden. Myöhemmin toimeksiantajaa informoitiin kehitystyön etenemisestä sekä varmistettiin tältä, että seuraava tapaaminen kannattaa järjestää vasta seuraavan vuoden puolella. Tammikuun alussa toimeksiantajalle ilmoitettiin mm. lisättyjen toimenpiteiden automaattiseen ajoittamiseen liittyvien toimintojen ja toimenpiteen lisäyksen yhteydessä suunnitellun automaattiseen asiakastietojen täyttöön liittyvän toiminnon valmistumisesta. Toimeksiantaja antoi ymmärtää perehtyvänsä toimintoihin syvemmin sitten kun ne ovat ”valmiimpia”. Myöhemmin toimeksiantajalta kysyttiin tarkempia tietoja aiemmasta toiminnanohjausjärjestelmästä, johon

vastattiin, että se oli palvelu, johon maksettiin kuukausimaksu ja johon toimeksiantajalla oli käyttäjätunnukset. Hän myös sanoi, että palvelu oli irtisanottu käyttämättömänä päättyneen vuoden lopussa. Kuun lopussa kysyttiin vielä kalenteritoiminnon yksittäisen toimenpiteen muokkaukseen tarkoitetun popup-ikkunan yksityiskohdista, johon saatiin vastaus. Lisäksi ilmoitettiin toimeksiantajan toivoman toimenpiteeseen liittyvän drag and drop-toiminnon kuulumisesta oletuksena kalenterimoduuliin. Helmi-maaliskuun 2020 transaktio on koottu taulukkoon 6.

## HELMIKUU-MAALISKUU 2020

***Taulukko 6. Transaktio toimeksiantajan kanssa marraskuun helmi-maaliskuun 2020 osalta.***

| | |
|---|---|
| 07.02.2020 | Kysymyssarja mm. lisättäviin liitetiedostoihin liittyen |
| 11.03.2020 | Tapaamiseen liittyvistä yksityiskohdista sopimista |
| 12.03.2020 | Tapaaminen toimeksiantajan kanssa |
| 24.03.2020 | Kysymys liittyen käytössä olleen työnhallintajärjestelmän palveluntarjoajaan |
| 27.03.2020 | Toimeksiantajalta saatiin vastaus edelliseen viestiin |

Toimeksiantajalle lähetettiin ensin 4 kysymystä liittyen mm. lisättäviin liitetiedostoihin, Selaa toimenpiteet-valikkovalintaan liittyvän toimenpidelistauksessa näytettäviin tietoihin sekä toimenpiteiden listausjärjestykseen. Vastausta ei saatu, mutta kysymyksiin liittyvät ongelmat ratkaistiin myöhemmin parhaaksi katsotulla tavalla. Myöhemmin todettiin kaikkien sovittujen toimintojen olevan valmiit, jonka perusteella toimeksiantajan kanssa sovittiin tapaaminen puhelimitse. Myöhemmin tapaamisen yksityiskohdista sovittiin sähköpostitse. Tapaamisen yhteydessä katselmoitiin toteutetut toiminnot sekä otettiin vastaan uudet vaatimukset. Toimeksiantaja lisäksi ilmoitti, että oli päättänyt vapaata lähdekoodia edustavan Odoo-toiminnanohjausjärjestelmän myöhemmästä käyttöönotosta. Kehitettävää järjestelmää toimeksiantaja suunnitteli käyttävänsä kuluvan vuoden loppuun asti. Tapaamisessa myös otettiin toimeksiantajalta allekirjoitus diplomityöprojektia varten vaadittavaan ohjaussuunnitelmaan. Kuun lopussa tiedusteltiin vielä toimeksiantajan aiemman työnhallintajärjestelmän palveluntarjoajasta, johon saatiin vastaus, että se oli Tribuni. Samana päivänä varmistettiin vielä, että se todella oli kodinkonehuoltoa tekeville yrityksille suunniteltu versio, johon saatiin myöntävä vastaus. Huhti-kesäkuun 2020 transaktio on koottu taulukkoon 7.

## HUHTIKUU-TOUKOKUU-KESÄKUU 2020

***Taulukko 7. Transaktio toimeksiantajan kanssa huhti-touko-kesäkuussa 2020.***

| | |
|---|---|
| 11.04.2020 | Ilmoitus diplomityöraportin kirjoittamisen vieneen aikaa ohjelmistoprojektilta |
| 10.05.2020 | Toimeksiantaja kysyi tilannetietoja |
| 10.05.2020 | Toimeksiantajalle vastattiin edelliseen viestiin |
| 06.06.2020 | Ilmoitus järjestelmän vaatimasta uudelleensuunnittelusta |
| 25.06.2020 | Kysymys liittyen uusien vaatimusten tekniseen toteutukseen |
| 26.06.2020 | Toimeksiantaja vastasi edelliseen viestiin |

Diplomityöraportin kirjoittaminen vei aikaa itse ohjelmistoprojektilta kuun alussa ja siitä ilmoitettiin toimeksiantajalle myöhemmin. Seuraava transaktio tapahtui vasta toukokuussa toimeksiantajan kysyessä tilannetietoja huomattuaan kehitettävän sovelluksen olevan muuttumaton edelliseen tarkasteluun verrattuna. Vastausviestissä kerrottiin diplomityöraportin kirjoittamisen edelleen vieneen käytettävissä olevaa aikaa ohjelmistoprojektilta. Kun ohjelmistokehitykseen palattiin, huomattiin, että tapaamisessa saatuihin uusiin vaatimuksiin liittyen tarvitaan uudelleensuunnittelua. Tästä ilmoitettiinkin toimeksiantajalle. Samalla kysyttiin, että sopiiko tälle, että uusiin vaatimuksiin ja löydettyihin virhetoimintoihin liittyvän kehitystyön ollessa valmis, tälle ilmoitetaan erikseen. Vastausta odotellessa kehitystyötä jatkettiin, kunnes nousi tarve kysyä toimeksiantajalta uusimpiin vaatimuksiin liittyvistä yksityiskohdista; tämä liittyi vuosihuoltosopimuskäyntien mahdolliseen ajoittumiseen päällekkäin kalenterissa niin, että osa peittyy toistensa alle. Tällä kertaa viestiin vastattiin. Toimeksiantaja ilmoitti seikan olevan merkityksetön, mutta tälle kuitenkin ilmoitettiin, että toimenpiteiden päällekkäin sijoittumista välttävä algoritmi oli itse asiassa jo ehditty kehittää valmiiksi. Heinä-elokuun 2020 transaktio on koottu taulukkoon 8.

## HEINÄ-ELOKUU 2020

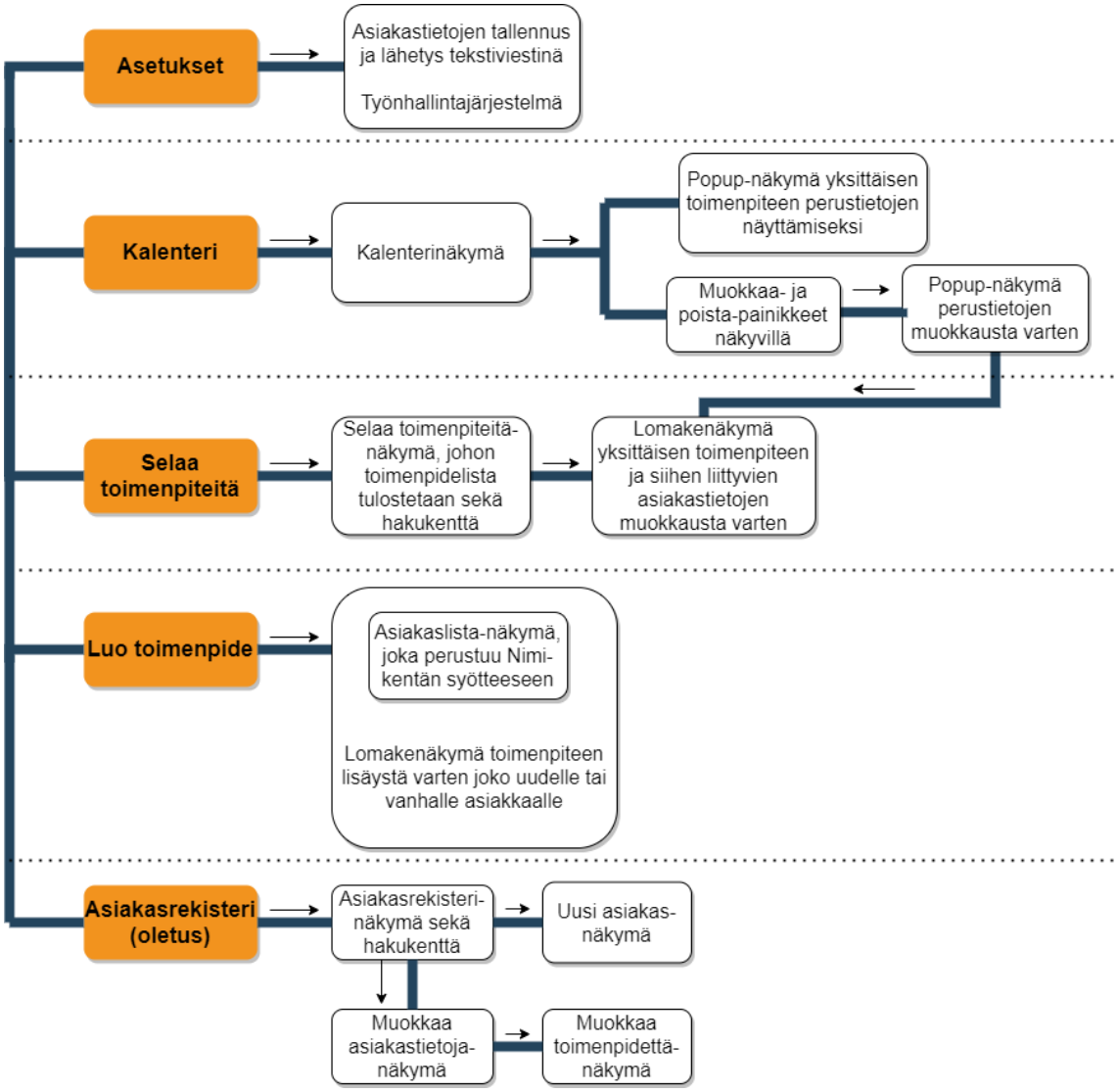
***Taulukko 8. Transaktio toimeksiantajan kanssa heinä-elokuussa 2020.***

| | |
|---|---|
| 31.07.2020 | Ilmoitus kaikkien toimintojen olevan valmiit |
| 27.08.2020 | Ilmoitus käyttöönottovalmiudesta |

Heinäkuun lopussa toimeksiantajalle ilmoitettiin, että kaikki sovitut toiminnot ja ominaisuudet ovat valmiit, mutta että käyttöönottoa varten tehtävät testit ovat vielä tekemättä. Projektin viimeinen transaktio tapahtui elokuun lopussa, jolloin ilmoitettiin myös käyttöönottoa varten suoritettujen testien olevan valmiit. Kun vastausta ei saatu, toimeksiantajalle soitettiin. Toimeksiantaja ilmoitti yrityksen tilausten lisäännyttyä huomattavasti aikaistaneensa Odoo-järjestelmän käyttöönottoa. Tämän perusteella kehitettävän järjestelmän käyttöönotto nähtiin tarpeettomaksi ja ohjelmistoprojekti päätettiin tähän.

## 4.2 Näkymät

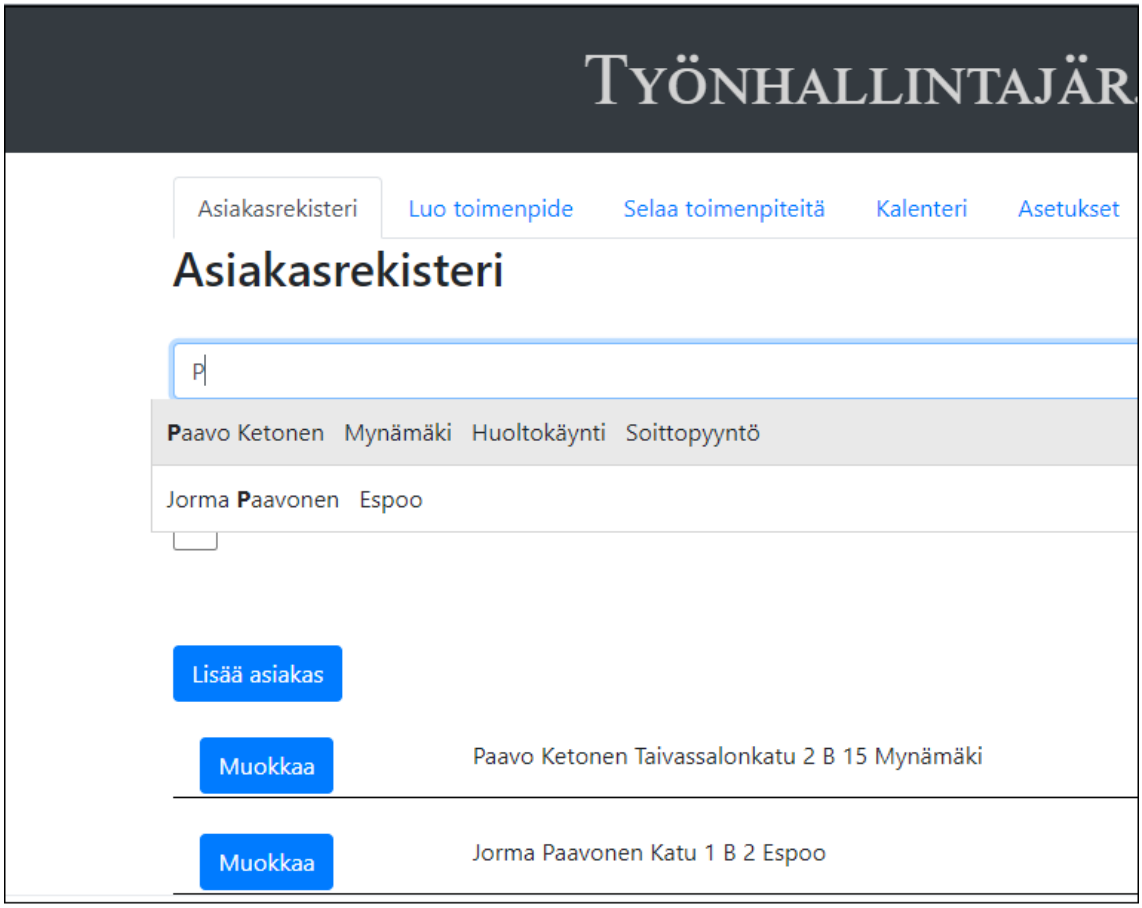
Toteutettuun sovellukseen toteutettiin kuvassa 8 kuvatut näkymät. Kuvaan on merkitty myös, kuinka näkymien välillä on mahdollista navigoida. Kaikki **Kalenteri**-osion näkymät toimivat niin, että päänäkymä on aina taustalla. Pienemmät näkymät siis avautuvat sen yhteydessä.




***Kuva 8.*** *Sovelluksen käyttöliittymän sivukartta ja näkymät.*

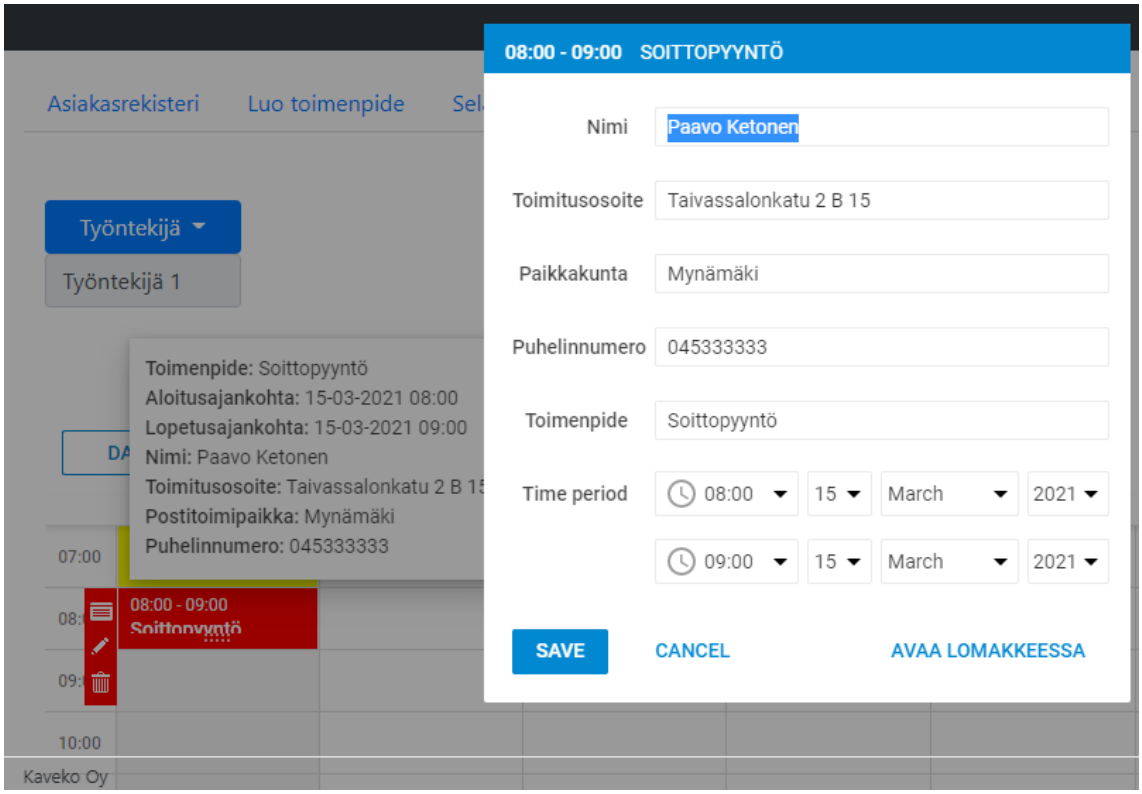
Hieman vastaavasti toimii myös **Luo toimenpide**-osion nimi-kentän perusteella generoitava vanhoista asiakkaista koostuva **Asiakaslista**-näkymä. **Selaa toimenpiteitä**- ja **Asiakasrekisteri**-näkymissä on hakukenttä, joka kummassakin toimii niin, että sen kautta voi tehdä haun asiakkaan nimen, paikkakunnan tai toimenpidetyypin perusteella.

Hakutuloksien perusteella generoidaan vastaava lista kuin **Luo toimenpide**-osion **Asiakaslista**-näkymässä sillä erolla, että viimeksi mainitun näkymän listaan generoidaan vain asiakkaan nimi – ei paikkakuntaa ja toimenpidetyyppiä kuten **Asiakasrekisteri**- ja **Selaa toimenpiteitä**-näkymissä. Esimerkki **Asiakasrekisteri**-näkymän hakukentän toiminnoista on esitetty kuvassa 9, jossa kuvitteelliselle asiakkaalle on osoitettu sekä *Huoltokäynti*-, että *Soittopyyntö*-tyyppiset toimenpiteet.




***Kuva 9.*** *Asiakasrekisterinäkymän hakukenttätoiminto.*

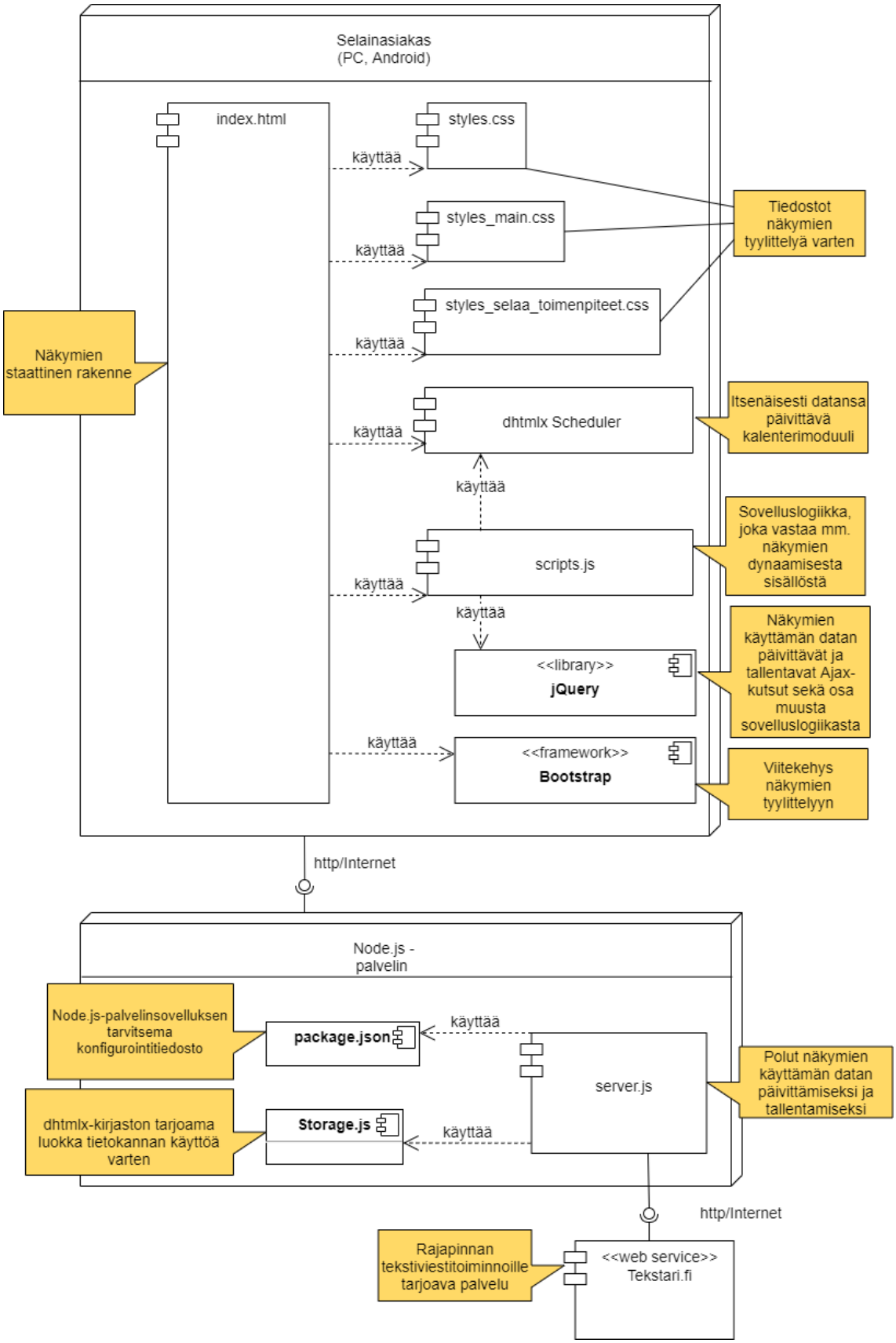
**Kalenterinäkymä**n kautta avautuva yksittäisen toimenpiteen perustietojen näyttämiseen tarkoitettu **Popup-näkymä** on esitetty kuvassa 10. Kyseisen näkymän kaikki muut kentät paitsi toimenpiteen aloitus- ja lopetusajat ovat muokattavissa. **Popup-näkymä**n kautta on pääsy **Selaa toimenpiteitä**-osion lomakenäkymään tarkasteltavan toimenpiteen tiedot lomakkeelle generoituna Avaa lomakkeessa-tekstiä klikkaamalla. Näkymissä navigointiin liittyvät tärkeimmät funktiokutsut on esitetty liitteessä C.




***Kuva 10.*** *Kalenterinäkymän kautta avautuva Popup-näkymä.*

## 4.3 Rakenneosat

Kuvassa 11 on esitetty sovelluksen moduulirakenne. Tiedosto scripts.js sisältää lähes koko sovelluslogiikan, jonka vuoksi aliluvussa 4.2 kuvatuissa näkymissä navigointiin liittyvät funktiokutsut ja tilatiedot keskittyvät kyseisen funktion koodiin. Node.js-ympäristössä ajettava palvelinkoodi vaatii lisäksi konfigurointitiedoston package.json. Sovelluksen Setup Node.js *App* suorittama komento ***npm install*** saa lisäksi aikaan kansion node_modules luomisen, jolloin paketinhallintasovellus lataa tiedostossa package.json määritellyt moduulit.



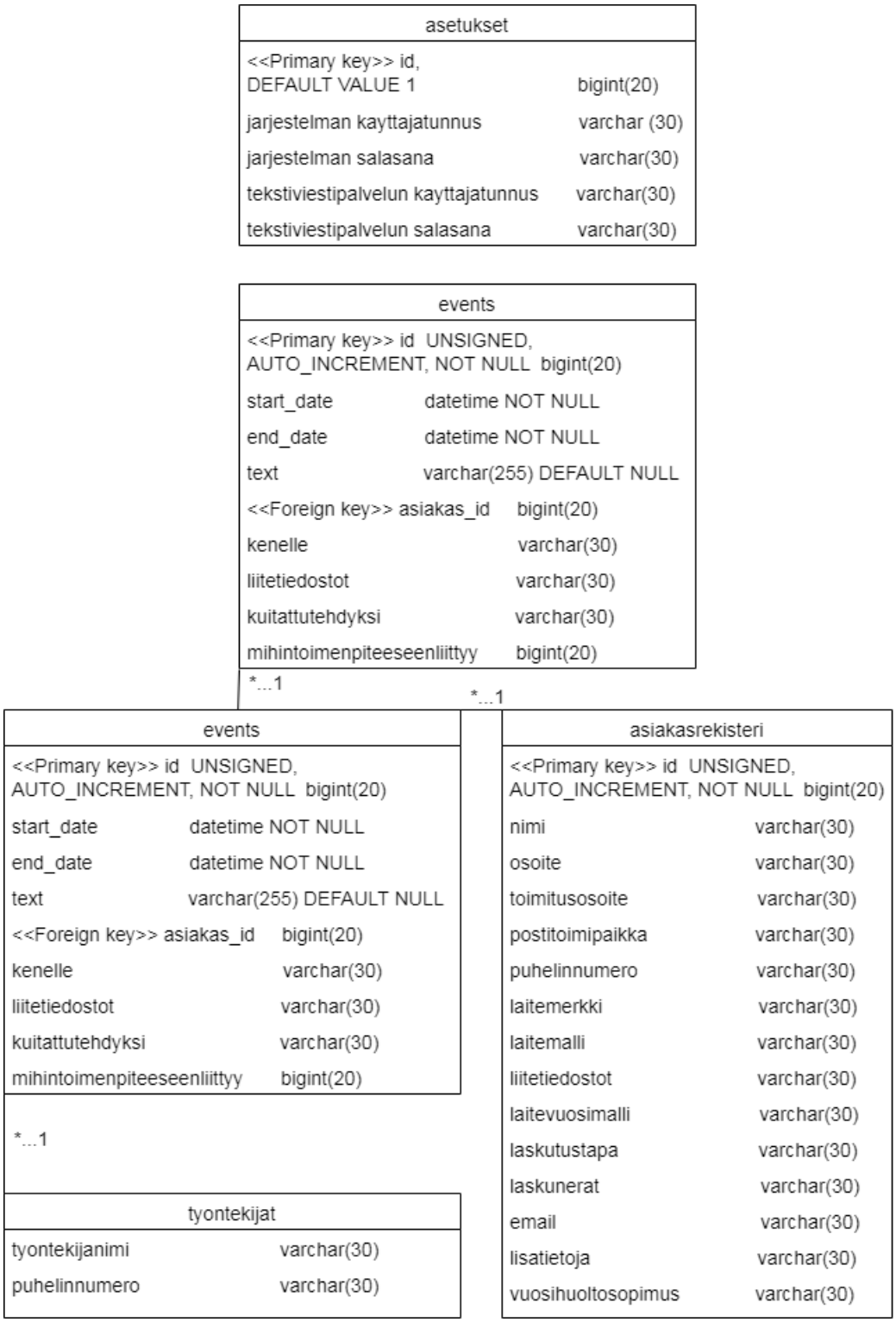

***Kuva 11.*** *Sovelluksen moduulirakenne.*

## 4.4 Toteutetun sovelluksen käsittelemät tiedot

Näkymissä näytettävä tieto perustuu tietokannasta ladattuun dataan, josta muodostetaan selainkoodissa 5 erilaista globaalia muuttujaa. Näistä *customersFromTheServer* on kaikki asiakkaat sisältävä objektilista. Myös *events* on objektilista ja se sisältää asiakkaiden tilaamat toimenpiteet asiakastietoineen, ja se perustuu palvelimelle lähetetyn työntekijänimen perusteella palautettuun toimenpidelistaan – se ei siis sisällä kaikkia toimenpiteitä. Sitä varten on erillinen objektilista *allEvents*. Kyseiset toimenpidelistat eivät myöskään vastaa tietokantataulun *events* tietoja, sillä toimenpidelistojen objektit sisältävät myös kuhunkin toimenpiteeseen liittyvät asiakastiedot. Näille toimenpide- ja asiakastiedoista koostuville entiteeteille on perustana **Luo toimenpide**-osion lomake, jossa toimenpiteelle lisätään asiakastiedot. Samantyyppistä lomaketta käytetään myös **Selaa toimenpiteitä**-osion muokkausnäkymässä. Myös työntekijät ja niiden puhelinnumerot sisältävä *tyontekijat* ja autentikointi- sekä tekstiviestipalvelun asetukset sisältävä *settings* ovat objektilistoja – viimeksi mainitun listan pituus voi tosin olla vain 1.

Sovelluksen käyttämään MYSQL-tietokantaan toteutetut tietokantataulut on esitetty kuvassa 12. Tauluista *events* on kalenterimoduuli Schedulerin määrittelemä ylhäältä laskien neljän ensimmäisen attribuutin osalta. Attribuuttia *text* käytetään toimenpidetyypin tallentamiseen, *kenelle* on varattu työntekijänimelle, jolle toimenpide on osoitettu sekä *kuitattu tehdyksi* ja *mihintoimenpiteisiinliittyy* liittyvät suoritetuksi kuitatun toimenpiteen automaattiseen vuoden päähän generointiin niin, että suoritettu toimenpide saa attribuutin *kuitattu tehdyksi* arvoksi suorituspäivämäärän, josta laskien vuoden päähän generoitava uusi toimenpide saa attribuutin *mihintoimenpiteeseenliittyy* arvoksi valmiiksi kuitatun toimenpiteen *id*-attribuutin arvon. Mainittu ominaisuus on rajattu vain toimenpidetyyppiin *Vuosihuoltosopimuskäynti*. Vierasavaimella *asiakas_id* toimenpide liittyy aina johonkin asiakkaaseen.

Sovelluksen toiminta perustuu useaan muuhunkin globaaliin muuttujaan, joista keskeisimpiä ovat **Muokkaa asiakastietoja**-näkymän tilaan liittyvä *isModifyState*, saman näkymän kautta avautuvan **Muokkaa toimenpidettä**-näkymän tilaan liittyvä *isModifyToimenpideOfAsiakaslomakeState* sekä **Selaa toimenpiteitä**-osion yksittäisen toimenpiteen muokkaukseen tarkoitetun näkymän tilaan liittyvä *selaaToimenpiteetModifyState*. Näiden lisäksi globaaleilla muuttujilla hallitaan muun muassa lomakenäkymiin liittyviä lisättäviä (*fileNames*), aiemmin lisättyjä (*earlierFileNames*) sekä poistettavia (*removables*) tiedostonimiä. Mainittujen muuttujien käsittelyä on esitetty liitteessä C.




***Kuva 12.*** *Työnhallintajärjestelmän tietokantataulut.*

# 5. PROJEKTIN ARVIOINTI

Tässä luvussa arvioidaan itse ohjelmistoprojekti, sen tuottama ohjelmisto ja kehittämiseen käytetyt työkalut kehitysmetodologian osana. Kahdessa ensimmäisessä aliluvussa käydään läpi toteutettua ohjelmistoprojektia verraten sitä standardin ISO/IEC TR29110 osan 5-1-1 ohjeisiin. Ohjeistuksen vaatima ja projektin lähtökohtana oleva kuvaus työstä saatiin projektin ensimmäisessä tapaamisessa toimeksiantajan kanssa, jossa luonnosteltiin tarvittavan sovelluksen käyttöliittymän toimintoja ja sen käsittelemää dataa. Kehitystiimi koostui yhdestä ainoasta henkilöstä eli diplomityöntekijästä, joka siis toimi sekä projektipäällikkönä, että itse kehitystiiminä. Ensimmäisessä aliluvussa verrataan toteutetun projektin projektinhallintaprosessia mainitun standardin vastaavaan. Toisessa aliluvussa taas saman standardin määrittelemää ohjelmiston toteuttamisprosessia verrataan toteutetun ohjelmistoprojektin vastaavaan. Kummankin aliluvun lopussa on prosessikohtainen taulukko, johon on koottu vertailun tulokset tehtäväkohtaisesti. Kolmannessa aliluvussa tarkastellaan vastaavaa taulukointia hyödyntäen tuotettua ohjelmistoa sen laajennettavuuden, koodin uudelleenkäytettävyyden ja sen luettavuuden osalta. Vastaavaa taulukointia käytetään myös käytettyjen työkalujen tarkasteluun. Viimeisessä aliluvussa pyritään etsimään projektin heikkouksiin johtaneita juurisyitä.

## 5.1 Toteutunut projektinhallintaprosessi

### PROJEKTISUUNNITTELU

Projektisuunnittelu – aktiviteetin arviointi on esitetty taulukossa 9. Toteutetussa ohjelmistoprojektissa ei tehty tähän aktiviteettiin kuuluvaa varsinaista projektisuunnitteludokumenttia, vaan dokumentointia tehtiin lähinnä ohjelmiston toteuttamiseksi laadittavien tehtävien osalta Scrum-tyyppisesti taulukkoa seuraavan kuvauksen mukaan.

***Taulukko 9. Projektisuunnittelu - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Project planning | 3 | 4 | 2 | 1 | 1 |
| PM.1.1 Työn kuvauksen katselmointi | | | | x | |
| PM.1.2 Tarvittavien tehtävien luonti | | x | | | |
| PM.1.3 Tehtävien keston arviointi | | x | | | |
| PM.1.4 Resurssien tunnistus ja dokumentointi | | | x | | |
| PM.1.5 Kehitystiimin muodostaminen | | | | | x |
| PM.1.6 Projektin kustannusten ja työmäärän arviointi | | x | | | |
| PM.1.7 Tehtävien aloitus- ja lopetuspäivämäärien määritys | x | | | | |
| PM.1.8 Projektin riskien tunnistus ja dokumentointi | x | | | | |
| PM.1.9 Koosta aiemmat vaiheet projektisuunnitelmaksi | x | | | | |
| PM.1.10 Katselmoi ja hyväksytä projektisuunnitelma | | | x | | |
| PM.1.11 Perusta projektikansio | | x | | | |

Työn kuvauksena toiminut toimintokokonaisuus koostettiin projektin ensimmäisessä tapaamisessa toimeksiantajan kanssa aluksi kynä-paperi-tekniikalla. Ne luokiteltiin asiakasvaatimuksiksi ja niitä koottiin myöhemmin listaksi liitteessä A esitetyn mukaisesti. Mainitussa liitteessä kuvattua asiakasvaatimusten UML-mallinnusta tehtiin vasta kun ohjelmisto oli valmis luovutettavaksi. Työn kuvausta katselmoitiin (PM 1.1) [12, s. 18] projektin aikana useastikin ja se liittyi yleensä vaatimusmäärittelyyn. Sitä ei kuitenkaan erikseen dokumentoitu, vaan paperiluonnokset ja sähköpostiviestihistoria toimi taltiointimuotona. Seuraavana tehtävälistassa oleva projektin tuotoksien aikaansaamiseksi laadittavien tehtävien tunnistusta (PM 1.2) tehtiin projektin alussa käytössä olleen GanttProject-sovelluksen avulla. Kyseinen sovellus toimi myös tehtävien dokumentointivälineenä. Varsinaisia ohjelmistokomponentteja ei kuitenkaan tunnistettu – eräänlaisina abstraktioina tosin pidettiin sovelluksen käyttöliittymän valikkovalintoja, joihin tarvittavat toiminnot oli tarkoitus toteuttaa. Standardia tulkittiin niin, että projektisuunnitelma ja siihen kuuluva tehtävien tunnistus ei ole välttämätöntä saada valmiiksi heti projektin alussa, vaan niitä tarvittaessa päivitetään niin kauan kuin projektissa ei ole vielä siirrytty projektin päättävään aktiviteettiin.

Tässäkin ohjelmistoprojektissa tehtäviä tunnistettiin niin kauan kuin sovellus oli keskeneräinen. Myöhemmin projektissa luovuttiin GanttProject-sovelluksesta ja siirryttiin Scrum-viitekehystä tukevan Asana-sovelluksen käyttöön. Myös sitä hyödynnettiin tehtävien tunnistamisessa ja taltioinnissa. Asana-sovellusta käytettiin tehtäväkirjanpidossa projektin viimeiseen tapaamiseen asti, jossa uudet vaatimukset kirjattiin Word-dokumenttiin ja tehtäväkirjanpitoa jatkettiin kyseisessä muodossa. Erillisiä tarkistus-, hyväksyntä- ja katselmointitehtäviä ei kuitenkaan tehty.

Kullekin kehitysprojektin tehtävälle tehtävässä keston arvioinnissa (PM 1.3) [12, s. 18] käytettiin aluksi GanttProject-sovellusta, mutta kun siirryttiin käyttämään puhtaasti Scrum-viitekehystä Asana-sovellusta käyttäen, tehtävien keston arvioinnista luovuttiin. Tehtävään PM.1.4 vaatimaan resurssien tunnistamiseen ja dokumentointiin liittyen tunnistettiin ensinnäkin webhotellin Node.js-ympäristö sovelluksen kehitysvaiheiden visualisoimiseksi. Muita tunnistettavia resursseja itse ohjelmistoprojektin kannalta olivat diplomityöntekijän ohjelmistokehitykseen käyttämä laitteisto erikseen luotavine lokaalipalvelimineen ja koodieditoireineen sekä Android-puhelin sovelluksen responsiivisuusvaatimuksiin liittyvää testausta varten. Resursseja ei kuitenkaan dokumentoitu. Henkilöresursseiksi tunnistettiin jo diplomityöprojektin puitteissa paitsi itse diplomityöntekijä, myös toimeksiantajan edustaja sekä diplomityönohjaaja ja ne dokumentoitiin.

Tehtävien alustavia aloitus- ja lopetusaikoja taas asetettiin vielä projektin alkuvaiheessa GanttProject-sovelluksen ollessa käytössä (PM.1.6) [12, s. 18], mutta kun siirryttiin käyttämään Scrum-perustaista Asanaa, siitä luovuttiin. Tehtävään PM.1.7 liittyvää työmäärän arviointia ei tehty missään vaiheessa. Samaan tehtävään liittyvää kustannusarvioita ei myöskään nähty tarpeelliseksi, koska kyseessä oli täysin opiskelijavetoinen projekti, jossa sovittiin vain mahdollisten matka- ym. kulujen korvaamisesta.

Tehtävän PM.1.8 [12, s. 19] vaatimaa riskien tunnistusta ja dokumentointia ei tehty, mutta toimeksiantajan fyysinen etäisyys tiedostettiin eräänlaiseksi riskiksi projektin sujuvuuden kannalta. Tehtävään PM.1.9 vaatimaa aiempien tehtävien koostamiseksi yhtenäiseksi projektisuunnitelmaksi ei tehty, mutta tehtävään PM.1.10 vaatimaa projektisuunnitelman hyväksytystä toimeksiantajalla tehtiin siltä osin, että aloituspalaverissa hyväksytettiin käyttöliittymän layout, Node.js-tuella varustetun webhotellin käyttöönotto sovelluksen valmistuttua sekä suunnitelma valmistuneiden toimintojen katselmoinnista suoraan diplomityöntekijän webhotelliin kehitettävän sovelluksen avulla. Suunnitelmien hyväksytystä tehtiin myöhemmin myös erillisissä sprinteissä vaatimuksista toteutustasolle edetessä, jolloin nousi esiin välttämättömiä teknisiä yksityiskohtia, joiden toteutustavasta kysyttiin mielipidettä toimeksiantajalta. Tehtävässä PM.1.11 vaadittava projektikansion perustaminen toteutettiin siten, luotiin kaksi erillistä kansiota ja niitä käytettiin projektin aikana kokeiltavien ulkoisten ohjelmistomoduulien toimintojen tarkasteluun, lokaalipalvelimella tapahtuvaan kehitystyöhön ja itse diplomityöhön liittyvän materiaalin taltioimiseen.

## SUUNNITTELUN TOIMEENPANO

Suunnittelun toimeenpano – aktiviteetin arviointi on esitetty taulukossa 10. Tehtävässä PM.2.1 [12, s. 19] vaadittavaa projektin tilan seurantaa toteutettiin ensin GanttProject-sovellusta hyödyntäen.

***Taulukko 10. Suunnittelun toimeenpano - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Project plan execution | 0 | 1 | 1 | 0 | 0 |
| PM.2.1 Projektisuunnitelman toteutumisen seuranta ja taltiointi | | | x | | |
| PM.2.2 Tapaamisten sopiminen toimeksiantajan kanssa | | x | | | |

Kyseisessä sovelluksessa on mahdollista asettaa tehtäville toteutumisprosentti. Toimintoa käytettiinkin kyseisen sovelluksen käytön ajan. Myöhemmin projektin tilaa seurattiin vain Asana-sovelluksen tehtävien valmiiksi kuittauksien ja Word-muodossa olleiden tehtävien yliviivauksien avulla. Aktiviteetin tehtävässä PM.2.2 [12, s. 20] vaadittavaan asiakastapaamisten toteutukseen liittyen ei ollut paljoa liikkumavaraa osapuolten fyysisen etäisyyden takia, mutta kun niitä järjestettiin (projektin aikana yhteensä 2), sovitut uudet

vaatimukset taltioitiin sekä niiden toteuttamista seurattiin ohjelmistoprojektin päätökseen asti. Yhden muutospyynnöksi tulkitun lisävaatimusjoukon toteuttamisesta johtuvasta lisääntyneestä työmäärästä huolimatta niistä ei neuvoteltu, vaan ne hyväksyttiin sellaisenaan. Myös tapaamisten taltiointi ja muutospyyntöihin liittyvät neuvottelut kuuluvat tehtävän PM.2.2 vaatimuksiin.

## PROJEKTIN ARVIOINTI JA KONTROLLOINTI

Projektin arviointi ja kontrollointi – aktiviteetin arviointi on esitetty taulukossa 11. Vertailua luotujen tehtävien toteutumisesta projektisuunnitelmaan nähden (PM.3.1, 1. osa) [12, s. 20] tehtiin verraten GanttProject- ja Asana-sovellusten kautta taltioitujen tehtävien toteutumista.

***Taulukko 11. Projektin arviointi ja kontrollointi - aktiviteetin arviointi***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Project assessment and control | 0 | 1 | 0 | 2 | 0 |
| PM.3.1 Projektin etenemisen arviointi | | x | | | |
| PM.3.2 Arvioi muutospyyntöjä ja seuraa niiden toteutumista | | | | x | |
| PM.3.3 Poikkeusten korjaus ja toteutumisen seuranta | | | | x | |

Myös niiden suoritusjärjestystä pyrittiin tarvittaessa priorisoimaan uudelleen. Samoin tehtiin myös projektin myöhemmässä vaiheessa Word-muotoon tallennettujen tehtävien osalta. Kaikki toiminnalliseen määrittelyyn (kts. tilakaaviot liite B) suoraan liittyvät tehtävät lukuun ottamatta joitakin virheilmoituksiin liittyviä toimintoja kuitattiin tehdyksi ohjelmistoprojektin loppuun mennessä. Eri toimintojen protoiluvaiheissa luotiin tehtäviä ja osa protoista ei edennyt lopulliseen toteutukseen asti.

Tehtävän PM.3.1 toisessa osassa ohjeistettua arviota toteutuneiden resurssien kohdentamisesta suunniteltuihin tehtiin siltä osin, että projektin viimeisessä tapaamisessa havahduttiin, että Android-testausta ei oltu tehty vielä siinä vaiheessa projektia lainkaan. Projektin alussa tosin suunniteltiin, että sen osuuden voi jättää selaimella tehtyä testausta pienemmäksi Bootstrap-kirjaston käytön ansiosta. Kolmannessa osassa mainittua toteutuneen budjetin vertailua suunniteltuun ei tehty, koska projektille ei ollut tarvetta luoda budjettia. Neljänteen osaan liittyvä suunnitellun aikataulun vertailu toteutuneeseen oli projektiryhmällä huolenaiheena varsinkin, kun projekti oli noin kuukauden tauolla. Jääminen jälkeen aikataulusta siis tiedostettiin, mutta sitä ei dokumentoitu. Kun toimeksiantaja antoi muutospyynnön lisävaatimusten muodossa noin kaksi kuukautta ennen projektin alkuperäistä takarajaa projektin viimeiseksi jääneessä tapaamisessa, tämän kanssa sovittiin, että sovelluksen ei tarvitse olla valmis alkuperäisessä aikataulussa. Uutta aikataulua ei kuitenkaan määritelty tarkasti. Tehtävän viimeisen osan (tunnistettujen riskien vertailu toteutuneisiin) suhteen todettiin, että visualisoimalla toimeksiantajalle

nopeaan tahtiin sovellusluonnoksia sovituista ominaisuuksista ja toiminnoista sekä pitämällä aktiivisesti yhteyttä häneen vaatimusten täsmentämiseksi fyysiseen etäisyyteen liittyvä riski oli saatu minimoitua. Myöhemmin projektissa tapahtuneen tauon jälkeen toimeksiantajaa informoitiin projektin statuksesta varsinkin merkittävän edistymisen tapahduttua. Yhteydenpito tapahtui kuitenkin lähinnä sähköpostitse, johon toimeksiantaja ei ilmeisesti aina ehtinyt vastaamaan.

Kun viimeiset sovitut toiminnallisuudet olivat lähes valmiit, toimeksiantajaan saatiin yhteys puhelimitse projektin viimeiseksi jäänyttä tapaamista varten. Tapaamista varten kehitettävään sovellukseen toteutettiin viimeiset sovitut toiminnallisuudet, mutta niiden testaukseen ei jäänyt riittävästi aikaa, jonka vuoksi laadunvarmistukseen (testaus) jäi liian vähän aikaa. Siitä johtuen, tapaamisessa huomattiin, että osa toiminnoista toimi virheellisesti. Virhetoimintojen löytyminen oli aluksi pettymys toimeksiantajalle, mutta hänet saatiin vakuutettua, että ne korjataan ja että myös uudet vaatimukset toteutetaan. Toimeksiantaja oli kuitenkin löytänyt vaihtoehdon alusta asti toteutettavalle työnhallintajärjestelmälle. Kyseessä on Odoo-niminen vapaalla lähdekoodilla toteutettu järjestelmä, jonka käyttöönottoon yritys suunnitteli siirtyvänsä viimeistään seuraavana vuonna. Kehitteillä olevalle järjestelmälle toimeksiantaja arveli olevan käyttöä saman vuoden loppuun asti.

Tehtävässä PM.3.2 [12, s. 20] arvioidaan toimeksiantajalta saatuja muutospyyntöjä ja niiden toteutumista. Se toteutui niin, että projektin viimeisessä tapaamisessa saatua, muutospyynnöiksi tulkitun vaatimuslistan perusteella toteutettujen ominaisuuksien toteutumista arvioitiin ja seurattiin siten, että niiden perusteella luotuja tehtäviä toteutettiin vastaavasti kuin muidenkin tehtävien toteutumista. Tehtävään PM.3.3 kuuluvat toimet havaituista poikkeamista ja ongelmista toteutuivat lähinnä siten, että kun tehtiin mitä tahansa ohjelmistotestausta projektin aikana tai muutoin havaittiin poikkeamia tavoiteltuihin toimintoihin, niin niiden korjaamiseksi luotiin erilliset tehtävänsä. Tämä toteutettiin niin, että havaitut poikkeamat kirjattiin Word- tai Asana-sovelluksen määrittelemässä muodossa ja niiden pohjalta luotiin uusia tehtäviä poikkeamien korjaamiseksi.

## PROJEKTIN PÄÄTÖS

Projektin päätös – aktiviteetin arviointi on esitetty taulukossa 12. Koko projektinhallintaprosessin arvioinnin yhteenveto on taulukossa 13.

***Taulukko 12. Projektin päätös - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Project closure | 0 | 0 | 0 | 0 | 2 |
| PM.4.1 Projektin päätöksen formalisointi | | | | | x |
| PM.4.2 Projektikansion päivittäminen | | | | | x |
| Projektin päätös – aktiviteetti jäi kokonaan pois koska tuotetta ei luovutettu | | | | | |

***Taulukko 13. Projektinhallintaprosessin arviointi - yhteenveto.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Project Management Process | 3 | 6 | 3 | 3 | 3 |
| PM.1.1 Työn kuvauksen katselmointi | | | | x | |
| PM.1.2 Tarvittavien tehtävien luonti | | x | | | |
| PM.1.3 Tehtävien keston arviointi | | x | | | |
| PM.1.4 Resurssien tunnistus ja dokumentointi | | | x | | |
| PM.1.5 Kehitystiimin muodostaminen | | | | | x |
| PM.1.6 Projektin kustannusten ja työmäärän arviointi | | x | | | |
| PM.1.7 Tehtävien aloitus- ja lopetuspäivämäärien määritys | x | | | | |
| PM.1.8 Projektin riskien tunnistus ja dokumentointi | x | | | | |
| PM.1.9 Koosta aiemmat vaiheet projektisuunnitelmaksi | x | | | | |
| PM.1.10 Katselmoi ja hyväksytä projektisuunnitelma | | | x | | |
| PM.1.11 Perusta projektikansio | | x | | | |
| PM.2.1 Projektisuunnitelman toteutumisen seuranta ja taltiointi | | | x | | |
| PM.2.2 Tapaamisten sopiminen toimeksiantajan kanssa | | x | | | |
| PM.3.1 Projektin etenemisen arviointi | | x | | | |
| PM.3.2 Arvioi muutospyyntöjä ja seuraa niiden toteutumista | | | | x | |
| PM.3.3 Poikkeusten korjaus ja toteutumisen seuranta | | | | x | |
| PM.4.1 Projektin päätöksen formalisointi | | | | | x |
| PM.4.2 Projektikansion päivittäminen | | | | | x |

## 5.2 Toteutunut sovelluskehitysprosessi

### PROJEKTIN ALOITUS JA VAATIMUSTEN ANALYYSI

Projektin aloitus – aktiviteetin arviointi on esitetty taulukossa 14. Aktiviteetti toteutui yhden hengen projektissa vain tehtävän SI.1.2 [12, s. 27] osalta.

***Taulukko 14. Projektin aloitus – aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Software Implementation initiation | 0 | 0 | 0 | 1 | 1 |
| SI.1.1 Projektisuunnitelman katselmointi | | | | x | |
| SI.1.2 Kehitysympäristön luominen | | | | | x |

Tehtävässä vaaditaan joko kehittämisympäristön luomista tai sen päivittämistä projektia varten. Se toteutui niin, että webhotelliin asennettiin alustava versio kehitettävästä sovelluksesta. Kehittäjällä oli tarvittava välineistö kuten koodieditorit valmiina tietokoneellaan ja lokaalipalvelin pystytettiin vasta kun sitä tarvittiin kehitystyössä.

Vaatimusten analyysi – aktiviteetin arviointi on esitetty taulukossa 15. Sen tehtävässä SI.2.2 joko dokumentoidaan tai päivitetään toiminnallinen määrittely.

***Taulukko 15. Vaatimusten analyysi - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Software requirements analysis | 0 | 2 | 0 | 0 | 1 |
| SI.2.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.2.2 Toiminnallisen määrittelyn dokumentointi tai päivittäminen | | x | | | |
| SI 2.3 Hanki hyväksyntä toiminnalliselle määrittelylle | | x | | | |

Toteutuneessa projektissa kyllä koottiin mahdollisimman paljon vaatimuksia heti projektin alussa pitämällä tiiviisti yhteyttä toimeksiantajaan, mutta niiden dokumentointi rajoittui aluksi lähinnä sähköpostiviestien ja paperiluonnoksien muotoon. Vasta myöhemmin vaatimuksia koottiin listaksi liitteessä A esitetyn mukaan. Kyseisessä liitteessä esitettyä vaatimusten mallinnusta tehtiin osin jo projektin alussa, mutta pääosa siitä toteutettiin vasta itse ohjelmistoprojektin loppupuolella ja sen jälkeen. Asiakkaan esittämiä vaatimuksia myös arvioitiin ovatko ne asiakas- vai ohjelmistovaatimuksia. Suuri osa esitetyistä vaatimuksista arvioitiin lopulta asiakasvaatimuksiksi, sillä asiakkaalla oli selvä käsitys haluamaltaan järjestelmältä eikä tarkoituksenmukaisia vaihtoehtoisia tapoja esimerkiksi liitteessä A esitettyihin vaatimuksiin liittyen löydetty. Vaatimusmäärittelyä tehtiin lopulta koko projektin ajan, sillä varsinkin toiminnallinen määrittely tarkentui vähitellen sitä mukaa kun websovelluksen osioita – valikkovalintoja toiminnallisuuksineen toteutettiin.

Toteutetussa ohjelmistoprojektissa vaatimuksia ei siis koottu yhdeksi dokumentaatioksi kerralla ja siitä johtuen myöskään toiminnallista määrittelyä ei tehty tehtävässä kuvatulla tavalla. Suuri osa sovelluksen toiminnoista kuitenkin selvitettiin asiakasvaatimusten perusteella jo projektin alussa. Vaatimuksia nousi lopulta esiin lisää kehitysprojektin edetessä ja varsinkin loppupuolella, jolloin toimeksiantaja esitti ideoimansa lisätoiminnot järjestelmään. Dokumentoitu toiminnallinen määrittely laadittiin lopulta vasta itse ohjelmistoprojektin jälkeen. Kyseinen dokumentti on tämän diplomityön liitteessä B. Tehtävään kuuluva vaatimusten oikeellisuuden tarkistus toteutui inkrementaalisesti toteutetun projektin vaiheissa, joissa haluttiin varmistaa, että asiakkaan esittämät vaatimukset on ymmärretty oikein. Tämä tapahtui usein tilanteissa, joissa toimintoja jo kehitettiin ja sen yhteydessä nousi toteutustapaan liittyviä kysymyksiä, joiden katsottiin vaativan hyväksyntää asiakkaalta. Testattavuutta ei vielä tässä vaiheessa kuitenkaan pohdittu.

Tehtävässä SI.2.3 [12, s. 28] asiakas vastaa toiminnallisen määrittelyn hyväksynnästä ja sen oikeellisuuden vahvistamisesta. Koska yhtenäistä toiminnallista määrittelyä ei toteutettu vielä itse kehitysprosessin aikana toimintoja hyväksytettiin jopa toiminto kerrallaan visualisoimalla niitä webhotellissa toimivan websovelluksen avulla.

## OHJELMISTOKOMPONENTTIEN IDENTIFIOINTI

Ohjelmistokomponenttien identifiointi – aktiviteetin arviointi on esitetty taulukossa 16. Tehtävä SI.3.2 [12, s. 28] asettaa vastuun toiminnallisen määrittelyn ymmärtämisestä.

***Taulukko 16. Ohjelmistokomponenttien identifiointi - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Software component identification | 1 | 0 | 1 | 0 | 1 |
| SI.3.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.3.2 Ymmärrä vaatimusmäärittely | | | x | | |
| SI 3.3 Dokumentoi tai päivitä ohjelmistokomponenttien identifiointi | x | | | | |

Koska toteutuneessa ohjelmistoprojektissa ei tehty varsinaista toiminnallista määrittelyä vielä projektin alussa, kehitystiimillä oli kyseistä tehtävää vastaavassa vaiheessa selvillä kehitettävän sovelluksen toiminnoista lähinnä sen käyttöliittymän valikkovalintojen perustoiminnot. Näin ollen, kyseiseen tehtävään liittyvää kehitystiimille kuuluvaa toiminnallisen määrittelyn ymmärtämistä ei tehty standardin kuvaamalla tavalla, vaan toiminnallinen määrittely täsmentyi projektin aikana ja sen oikea ymmärtäminen varmistettiin tarvittaessa toimeksiantajalta kysymällä. Tehtävässä SI.3.3 joko dokumentoidaan tai päivitetään ohjelmistokomponenttien tunnistukseen liittyvät dokumentit. Se tapahtuu analysoimalla toiminnallista määrittelyä niin, että sen tuloksena ohjelmistokomponentit ovat selvillä niiden asetteluineen sekä ulkoisine ja sisäisine rajapintoineen. Tähän liittyen eräänlaisina abstraktioina ajateltiin siis käyttöliittymän valikkovalintoja ja palvelinpuolen sovellusta, joihin oli tarkoitus lähteä kehittämään toimintoja valikkovalinta kerrallaan, mutta varsinaisia komponenttitason hahmotelmia ei tehty. Myös tehtävän vaatima ohjelmistokomponentteihin liittyvä dokumentaatio jäi tekemättä. Projektin jälkeen laadittiin kuvaukset vain valmiin sovelluksen käyttöliittymän näkymistä sekä moduulirakenteesta.

## OHJELMISTON RAKENTAMINEN

Ohjelmiston rakentaminen – aktiviteetin arviointi on esitetty taulukossa 17. Koska komponenttitason arkkitehtuurisuunnittelua ei tehty, myöskään tehtävän SI.4.2 [12, s. 29] vaatimaa ohjelmistokomponenttien tunnistuksen ymmärtämistä ei toteutettu projektissa millään tavoin.

***Taulukko 17. Ohjelmiston rakentaminen - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Software construction | 1 | 3 | 0 | 0 | 1 |
| SI.4.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.4.2 Ymmärrä ohjelmistokomponenttien tunnistukseen liittyvät asiat | x | | | | |
| SI 4.3 Toteuta tai päivitä ohjelmistokomponentit | | x | | | |
| SI.4.4 Luo tai päivitä testitapaukset ja - proseduurit | | x | | | |
| SI.4.5 Testaa ohjelmistokomponentit | | x | | | |

Samasta syystä, myöskään tehtävään SI.4.3 liittyvää ohjelmistokomponenttien rakentamista ei tehty standardin määrittelemällä tavalla. Toimintoja siis kehitettiin pääasiassa valikkovalinta kerrallaan, mutta koska niiden data on yhteistä (toimenpide- ja asiakastiedot), niitä oli kehitettävä myös rinnakkain – lähinnä varmistaen, että data päivittyy kussakin valikkovalinnassa oikein senkin jälkeen, kun johonkin niistä on tehty muutoksia. Toimintoja toteutettaessa koodiin upotettiin jonkin verran kommentteja. Se kuitenkin koettiin viimeistään ohjelmointivaiheen lopussa riittämättömäksi koodin selkeyden ja ymmärrettävyyden kannalta.

Ohjelmiston rakentaminen aloitettiin **Luo toimenpide** - osiosta, jonka yhteydessä oli päätettävä myös käyttöönotettavasta tekstiviestipalvelusta. Palveluntarjoajaksi valikoitui lopulta kotimainen Tekstari.fi, joka tarjoaa rajapinnan (API) sovelluskehittäjien käyttöön. Mainitun valikkovalinnan toimintoja kehittäessä myös tietokannan ensimmäinen versio luotiin palvelinpuolen sovellukseen. Kyseessä on MYSQL-tietokanta, jota käytettiin aluksi suoraan ilman rajapintaluokkaa palvelinsovelluksen kautta. Seuraavaksi siirryttiin **Kalenteri** - osioon, jossa toimenpiteet oli tarkoitus tulostaa niiden ajankohtien mukaan. Sitä suunniteltaessa todettiin, että itse suunniteltuna ja toteutettuna se olisi ollut liian työläs projektiriskit huomioiden. Sen sijaan osion kehitystyössä päädyttiin kokeilemaan avoimella lähdekoodilla tuotettuja kalenterimoduuleja. Ensin kokeiltiin FullCalendar-nimistä moduulia, mutta sen integrointi järjestelmään koettiin epäselväksi. Sen sijaan Scheduler-moduulin dokumentaatioon oltiin tyytyväisiä ja se yhdessä kyseisen moduulin käytettävyyden kanssa johtikin sen valintaan. Moduuli määritteli lopulta tietokannan käsitteistöä ja toteutusta uusiksi sillä se tarjosi muun muassa *JavaScriptin* luokkasyntaksia noudattavan rajapinnan tietokannan käyttämiseksi Node.js-palvelinsovelluksen kautta.

Tehtävässä SI.4.4 määriteltyä testitapausten ja niiden proseduurien luomista sekä yksikkö- että integraatiotestausta varten sekä tehtävään SI.4.5 liittyvää itse yksikkötestausta ei tehty standardin määrittelemällä tavalla, vaan testausta tehtiin tilanteen mukaan suunnitellen sitä mukaa kun uusia toimintoja kehitettiin eikä niistä luotu varsinaisia testitapauksia eikä niitä dokumentoitu. Kun uusia toimintoja kehitettiin ja sovelluslogiikkaan tehtiin muutoksia, varmistettiin paitsi kulloinkin työn alla olleen valikkovalinnan toimintojen suunniteltu toiminnallisuus ja useimmiten myös muiden valikkovalintojen toiminta varmistettiin. Kaikki testit suoritettiin manuaalisesti käyttöliittymän reagointia seuraamalla kaikissa ajateltavissa olevissa tilanteissa sekä selaimen ja tarvittaessa lokaalipalvelimen konsoliin tulostamalla. Testauksessa hyödynnettiin myös webhotellin tietokantaan päivittyvien muutoksien seurantaa. Manuaalinen testaustapa vaati erityistä huolellisuutta ja oli odotettua työläämpää.

Ohjelmiston rakentamisessa ilmeni yksi merkittävää viivästystä aiheuttava koodivirhe, jota ei aluksi löydetty. Ennen juurisyyn löytymistä ehdittiin todeta, että Chrome- ja Firefox-selaimet toimivat siihen asti toteutetulla koodilla ratkaisevasti eri tavalla. Ennen selainten välisen poikkeavan toiminnan löytymistä virheen luultiin johtuvan mahdollisesta Android-alustan selainten JavaScript-moottoreiden erilaisesta tavasta suorittaa koodia. Juurisyyksi lopulta paljastui eräässä käyttöliittymän valikkovalinnan kautta avautuvassa lomakkeessa sijaitsevan *input*-tagin *type*-attribuutin arvo, joka oli *submit*, vaikka kaikissa muissa lomakkeissa sen arvoksi asetettiin *button*. Korjauksen jälkeen selainten välillä ei havaittu poikkeavia toimintoja ja myös Chromen Android-versio todettiin toimivan odotetusti. Android-alustalla ei tosin tehty sovelluslogiikan osalta yhtä kattavia testauksia kuin PC-selaimilla, mutta käyttöliittymän osien mukautuminen tarkoituksenmukaisella tavalla pienelle näytölle varmistettiin.

## OHJELMISTON KOKOONPANO JA TESTAUS

Ohjelmiston kokoonpano ja testaus – aktiviteetin arviointi on esitetty taulukossa 18. Tämä aktiviteetti ei ollut toteutetussa projektissa erillinen toimintonsa lukuun ottamatta projektin lopussa ennen suunniteltua käyttöönottovaihetta toteutettua testausta.

***Taulukko 18. Ohjelmiston kokoonpano ja testaus - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Software integration and tests | 1 | 3 | 0 | 0 | 2 |
| SI.5.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.5.2 Ymmärrä testitapaukset ja - proseduurit | | x | | | |
| SI 5.3 Ohjelmiston kokoonpano | x | | | | |
| SI.5.4 Suorita ohjelmistotestaus | | x | | | |
| SI.5.5 Korjaa löydettyjä vikoja kunnes testit läpäistään | | x | | | |
| SI.5.6 Sisällytä vaatimusmäärittely ja ohjelmisto konfiguraatioon | | | | | x |

Siitä ja varsinaisen komponenttirakenteen puuttumisesta johtuen aktiviteetin tehtäviä ei tehty standardin määrittelemällä tavalla. Esimerkiksi tehtävään SI.5.2 [12, s. 30] liittyvä testitapausten ja -proseduurien ymmärtäminen ei ollut erillinen toimintonsa koska kehityksen aikaiset testit suoritettiin heti niiden suunnittelun jälkeen. Samassa tehtävässä vaadittavaa testausympäristön luomista tai vaihtoehtoisesti päivittämistä tehtiin vastaavasti – tilanteen mukaan. Testausten dokumentointia ei tehty toimintoja kehitettäessä lainkaan. Myöskään tehtävään SI.5.3 liittyvää ohjelmiston kokoonpanoa ei komponenttirakenteen puuttuessa toteutettu. Edellä mainituista seikoista johtuen, paitsi yksikkötestaus, myös integraatiotestausvaihe (vastaa tehtävää SI.5.4) oli suoritettu eri osioiden valmistuttua. Siitä huolimatta järjestelmää päätettiin testata vielä erikseen ennen sen suunniteltua luovuttamista ja tällä kertaa päätettiin myös luoda dokumentoidut testitapaukset proseduureineen sekä dokumentoida myös testien hyväksytty läpäisy. Tämän

testausprosessin aikana löydettiin joitain virhetoimintoja, mutta niitä korjattiin, kunnes kaikki testit oli läpäisty. Tämä vastannee aktiviteetin tehtävää SI.5.5, jossa virheiden korjausta tehdään niin kauan, kunnes integrointitestaus on läpäisty. Manuaalinen testaus koettiin tässäkin odotettua työläämmäksi. Testit järjestelmän luovuttamista varten on työn liitteessä D.

## TUOTTEEN LUOVUTUS

Tuotteen luovutus – aktiviteetin arviointi on esitetty taulukossa 19. Koko sovelluskehitysprosessin arviointi on esitetty yhteenvetona taulukossa 20.

***Taulukko 19. Tuotteen luovutus - aktiviteetin arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Product delivery | 0 | 0 | 0 | 0 | 3 |
| PM.6.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| PM.6.2 Ohjelmistokonfiguraation ymmärrettävyyden katselmointi | | | | | x |
| PM.6.3 Tuotteen luovutus projektipäällikölle ja edelleen asiakkaalle | | | | | x |
| **Tuotteen luovutus – aktiviteetti jäi kokonaan pois koska tuotetta ei luovutettu** | | | | | |

***Taulukko 20. Sovelluskehitysprosessin arviointi - yhteenveto.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Software Implementation process | 3 | 8 | 1 | 1 | 9 |
| SI.1.1 Projektisuunnitelman katselmointi | | | | x | |
| SI.1.2 Kehitysympäristön luominen | | | | | x |
| SI.2.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.2.2 Toiminnallisen määrittelyn dokumentointi tai päivittäminen | | x | | | |
| SI 2.3 Hanki hyväksyntä toiminnalliselle määrittelylle | | x | | | |
| SI.3.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.3.2 Ymmärrä vaatimusmäärittely | | | x | | |
| SI 3.3 Dokumentoi tai päivitä ohjelmistokomponenttien identifiointi | x | | | | |
| SI.4.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.4.2 Ymmärrä ohjelmistokomponenttien tunnistukseen liittyvät asiat | x | | | | |
| SI 4.3 Toteuta tai päivitä ohjelmistokomponentit | | x | | | |
| SI.4.4 Luo tai päivitä testitapaukset ja - proseduurit | | x | | | |
| SI.4.5 Testaa ohjelmistokomponentit | | x | | | |
| SI.5.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| SI.5.2 Ymmärrä testitapaukset ja - proseduurit | | x | | | |
| SI 5.3 Ohjelmiston kokoonpano | x | | | | |
| SI.5.4 Suorita ohjelmistotestaus | | x | | | |
| SI.5.5 Korjaa löydettyjä vikoja, kunnes testit läpäistään | | x | | | |
| SI.5.6 Sisällytä vaatimusmäärittely ja ohjelmisto konfiguraatioon | | | | | x |
| PM.6.1 Jaa projektiryhmälle tehtävät kunkin roolin mukaisesti | | | | | x |
| PM.6.2 Ohjelmistokonfiguraation ymmärrettävyyden katselmointi | | | | | x |
| PM.6.3 Tuotteen luovutus projektipäällikölle ja edelleen asiakkaalle | | | | | x |

## 5.3 Työtulosten ja työkalujen tarkastelu

Projektin työtulosten ja niiden aikaan saamiseksi käytettyjen työkalujen arviointi on koottu taulukkoon 21. Käyttöönottoa varten tehty testaus (kts. liite D) oli prosessi, jonka aikana pyrittiin varmistamaan kehitettyjen toimintojen ja ominaisuuksien oikeellisuus asiakkaan esittämien vaatimusten kannalta. Tästä käytetään taulukossa 4 merkintää ERP1.

***Taulukko 21. Työtulosten ja työkalujen arviointi.***

| | N | P | L | F | n/a |
|---|---|---|---|---|---|
| Työtulosten ja työkalujen tarkastelu | 2 | 3 | 0 | 1 | 0 |
| ERP1 Vaatimusten toteutuminen | | | | x | |
| ERP2 Sovelluksen laajennettavuus | | x | | | |
| ERP3 Koodin uudelleenkäytettävyys | x | | | | |
| ERP4 Koodin luettavuus | x | | | | |
| KE Koodieditorin soveltuvuus kehittämismetodologiaan | | x | | | |
| LP Lokaalipalvelimen soveltuvuus kehittämismetodologiaan | | x | | | |

Koska kaikki testit lopulta läpäistiin, ei toteutetussa sovelluksessa havaittu poikkeamia toimeksiantajan vaatimuksiin nähden. Lopullisen version hyväksytystä toimeksiantajalla ei tehty tämän vetäydyttyä projektista ennen kuin järjestelmä oli testattu käyttöönottovalmiiksi. Näin ollen, mainittuun tapaan arvioida sovelluksen vaatimustenmukaisuutta liittyy epävarmuutta.

Sovelluksen laajennettavuutta arvioitiin mahdollisten varastonhallinnasta ja laskutuksesta vastaavien laajennusosien osalta (taulukossa ERP2). Laajennettavuus nähtiin liittyvän ensinnäkin asiakkaiden tilaamiin toimenpiteisiin ja niiden tyyppeihin. Lisäksi se nähtiin liittyvän myös suoritettujen toimenpiteiden **Kuitattu tehdyksi**-attribuuttiin. Niiden avulla nähtiin olevan mahdollisuus esimerkiksi päivittää varastosaldoja ja laskuttaa asiakasta. Toimenpide- ja asiakasobjektit ovat luvussa 4.4 kuvatun mukaisesti saatavissa mahdollisten laajennusmoduulien käyttöön niille varattujen globaalien objektilistojen kautta, joka nähtiin laajennettavuutta edistäväksi tekijäksi. Edellä mainitut seikat huomioiden laajennettavuus luokiteltiin lopulta juuri ja juuri tasoa P vastaavaksi.

Koodin uudelleenkäytön suhteen arvioitiin (taulukossa ERP3), että koska sovelluslogiikka toteutettiin suurimmaksi osaksi niin, että vain pieni joukko toteutetuista funktioista toteuttaa sellaisenaan useampaan kuin yhteen valikkovalinnan toimintoon liittyvän toiminnon, arvioitiin uudelleenkäytettävyys tasolle N. Toisaalta kuvassa 13 on esitetty esimerkki osasta funktiota, joka on toteutettu tietyn valikkovalinnan – tässä tapauksessa Asiakasrekisteri-osion lomakedatan lähettämiseen palvelimelle. Kyseisen valikkovalinnan lomake on kuitenkin sekä Uusi asiakas-, että asiakkaalle osoitetun toimenpiteen muokkaukseen tarkoitetun Muokkaa toimenpidettä-näkymän käytössä. Näin ollen, myös

kyseisessä kuvassa esitetty funktio vastaa molempien mainittujen näkymien datan lähettämisestä palvelimelle. Myös koodin luettavuus (taulukossa ERP4) arvioitiin samalle tasolle, sillä koodiin kirjoitettiin vain vähän kommentteja sen luettavuuden tueksi.

```
function sendAsiakasDataToServer() {

    var nimi = document.getElementById("luoAsiakasNimi");
    var osoite = document.getElementById("luoAsiakasOsoite");
    var toimitusosoite = document.getElementById("luoAsiakasToimitusOsoite");
    var postitoimipaikka = document.getElementById("luoAsiakasPostitoimipaikka");
    var puhelinnumero = document.getElementById("luoAsiakasPuhelinnumero");
    var email = document.getElementById("luoAsiakasSahkopostiosoite");
    var laitteenmalli = document.getElementById("laitteenmalli");
    var laitteenvuosimalli = document.getElementById("laitteenvuosimalli");
    var lisatietoja = document.getElementById("uusiAsiakasLisätietoja");


   var selectedDevice;
    if (document.getElementById('laitemerkki').checked) {
        selectedDevice = document.getElementById('laitemerkki').value;

    } else if (document.getElementById('laitemerkki2').checked) {
        selectedDevice = document.getElementById('laitemerkki2').value;

    } else if (document.getElementById('laitemerkki3').checked) {
        selectedDevice = document.getElementById('laitemerkki3').value;

    } else if (document.getElementById('laitemerkki4').checked) {
        selectedDevice = document.getElementById('laitemerkki4').value;

    } else if (document.getElementById('laitemerkki5').checked) {
        selectedDevice = document.getElementById('laitemerkki5').value;

    }
```

***Kuva 13.*** *Esimerkki kahteen toimintoon käytettävästä funktiosta.*

Yhdeksi suurimmista webhotellin koodieditoriin liittyvistä haasteista osoittautui webhotellin autentikoinnin vanhentuminen joissain tapauksissa jopa alle minuutissa. Tämän todettiin myöhemmin olevan yhteydessä tilanteisiin, joissa verkkoselaimen historiatietoja ei oltu tyhjennetty viikkoihin. Vaikka File manager - toiminnosta tiedettiin puuttuvan suuri osa varsinaisissa kehittäjien tietokoneille suunnitelluissa koodieditoreissa tutuista ominaisuuksista, hakutoiminnon puuttuminen koettiin yllättävän suureksi haasteeksi varsinkin selainpuolen ainoan *JavaScript*- tiedoston osalta, sillä suuri osa sovelluslogiikasta toteutettiin siihen. Myös näkymiä varten luodun ainoan HTML-koodin pituus tuotti haasteita hakutoiminnon puuttuessa. Yhdessä verkkoselaimen säännöllisen historiatietojen poiston kanssa File manager kuitenkin todettiin välttäväksi työkaluksi.

Lokaalipalvelimen käyttöön liittyvistä haasteista suurin oli webhotellissa toimivan palvelinkoodin ja lokaalipalvelimen koodin välinen synkronointi, jota tehtiin vain silloin kun lokaalipalvelimelle nähtiin välttämätöntä tarvetta. Tämä liittyi lähes aina tarpeeseen tulostaa palvelinkoodiin liittyviä virheilmoituksia tms. selaimen konsoliin, jota varten lokaalipalvelimen koodi oli synkronoitava webhotellin koodia vastaavaksi. Tällöin myös pyrittiin ratkaisemaan virheilmoituksen syy lokaalipalvelimelle kopioitua koodia muokkaamalla, jolloin syntyi tarve synkronoida webhotellin koodi lokaalipalvelinta vastaavaksi. Synkronoinnit toteutettiin tiedostonlatauksin. Synkronointitarpeet vaikuttivat arviointiin niin, että lokaalipalvelimen käytettävyys työssä käytetyn kehitysmetodologian kera nähtiin juuri ja juuri tasolle P kuuluvaksi.

## 5.4 Juurisyyarviointi

Prosessien aktiviteetit asetettiin merkitsevyysjärjestykseen arvioitujen heikkouksien osalta niin, että edellinen aktiviteetti on aina sen seuraajan juurisyyaktiviteetti. Näin ollen, listan ensimmäinen aktiviteetti on prosesseissa havaittujen heikkouksien juurisyyaktiviteetti.

1. Projektisuunnittelu
    - PM.1.2 ohjelmistokomponenttien luomiseksi laadittavien tehtävien puuttuminen
2. Ohjelmistokomponenttien identifiointi
    - SI.3.3 ohjelmistokomponenttien tunnistus, niiden suunnittelu rajapintoineen ja niiden dokumentointi
3. Ohjelmiston rakentaminen
    - SI.4.2 ohjelmistokomponentit hahmotettiin eri tavalla kuin standardissa
    - SI.4.3 ohjelmistokomponenttien rakentamista ei tehty standardin määrittelemällä tavalla
    - SI.4.4 dokumentoitujen testitapausten ja – proseduurien luominen toiminnalliseen määrittelyyn ja **Ohjelmistokomponenttien tunnistus** - aktiviteettiin perustuen jäi tekemättä
    - SI.4.5 ohjelmistokomponenttien testausta ei voitu tehdä standardin määrittelemällä tavalla, lisäksi manuaalinen testaus koettiin odotettua enemmän resursseja vaativaksi
4. Ohjelmiston kokoonpano ja testaus

- koko aktiviteetti - lisäksi manuaaliset testit koettiin odotettua enemmän resursseja vaativiksi.

Keskiössä on ohjelmistokomponenttien suunnittelu, johon liittyvät ensimmäiset tehtävät olisi standardin mukaan pitänyt laatia jo projektin alussa projektinhallintaprosessin **Projektisuunnittelu**-aktiviteetissa. Sen seurauksena ohjelmistokomponentteja ei identifioitu, suunniteltu rajapintoineen eikä dokumentoitu, vaan sovellusta kehitettiin lähinnä käyttöliittymän osio kerrallaan ainoana dokumentointitapana koodiin lisätyt kommentit. Ohjelmistoa ei siten myöskään rakennettu ohjelmistokomponentti kerrallaan eikä varsinaisia yksikkötestejä olisi edes voitu tehdä. Aktiviteetti **Ohjelmiston kokoonpano ja testaus** valittiin tuloksiin mukaan ensinnäkin siitä syystä, että ohjelmiston luovuttamista varten toteutettu manuaalinen testaus koettiin odotettua enemmän resursseja vaativaksi. Toiseksi, se valittiin tuloksiin mukaan syy-seuraus-suhteen perusteella; olihan jako komponentteihin jäänyt suunnitteluvaiheessa tekemättä, jonka vuoksi myöskään kokoonpanovaihetta ei tehty standardin mukaisesti.

Manuaalisen testauksen odotettua suurempi resurssitarve olisi pitänyt tämän projektin perusteella pitänyt huomioida riskiksi jo projektisuunnitteluvaiheessa. Jos kuitenkin tarvittavat tehtävät ohjelmistokomponenttien tunnistamiseksi olisi tehty, olisi todennäköisesti myös manuaalinen testaus vaatinut vähemmän resursseja. Manuaalista testausta ei kuitenkaan toteutetun projektin perusteella voi pitää ihanteellisena vaihtoehtona sen ohjelmallista testausta suuremman resurssitarpeen takia.

**Projektisuunnittelu**-aktiviteetin tehtävään PM.1.2 liittyvä ohjelmistokomponenttien luomiseksi laadittavien tehtävien puuttuminen arveltiin olevan juurisyynä myös työtulosten arviointiin liittyvien attribuuttien ERP2, ERP3 ja ERP4 vaatimattomaan luokitukseen; jako komponenteiksi nähtiin tekijäksi, joka olisi vaikuttanut jossain määrin ensinnäkin sovelluksen laajennettavuuteen kun lähdekoodi olisi jaettu tarkoituksenmukaisiin, uudelleenkäytön mahdollistaviin kokonaisuuksiin selkeine rajapintoineen. Jako selkeisiin komponentteihin itsessään olisi parantanut myös koodin luettavuutta. Itse jako komponenteiksi olisi ollut toteutettavissa *JavaScripti*n mahdollistaman oliosyntaksin avulla. Tällöin sekä selain-, että palvelinpään koodi olisi ollut jaettavissa useisiin eri tiedostoihin, jolloin myöskään File manager - hakutoiminnon puuttuminen ei olisi vaikuttanut yhtä paljon kehitystyön sujuvuuteen. Myös alan kirjallisuuslähde [8] painottaa sovellusten arkkitehtuurisuunnittelussa jaon komponenteiksi olevan tärkeää useimpien tässä aliluvussa käsiteltyjen ohjelmiston tavoiteltavien ominaisuuksien kannalta. Näitä ovat kyseisen lähteen mukaan ainakin muunneltavuus, ylläpidettävyys ja testattavuus.

# 6. YHTEENVETO

Diplomityön tulos on toimeksiantajan vaatimusten perusteella toteutettu SPA-malliin perustuva käyttöönottoa vaille valmis työnhallintajärjestelmä. Työn tuloksiin kuuluvat myös koko sovelluskehitysprojektin itsearviointi mukaan lukien kehitetty sovellus sekä sen toteuttamiseksi käytetyt työkalut kehitystyössä käytetyn metodologian yhteydessä. Projekti toteutettiin yhden henkilön toimesta etänä, jota varten kehitettiin metodologia inkrementaalisesti kehitettävän sovelluksen kehitysvaiheiden visualisoimiseksi toimeksiantajalle.

Kirjallisuuskatsauksen tärkeimmäksi hakutulokseksi metodologian kehittämisen kannalta osoittautui tutkimus [17], jossa kehitetään metodologia websovelluksen kehittämiseksi suoraan etäpalvelimelle kehittäjän tietokoneelta käsin käyttäen verkossa toimivia koodieditoreja koodin muokkaukseen ja verkossa toimivia ohjelmointiympäristöjä virheenetsintään. Kyseisten työkalujen käyttö ei tutkimuksen mukaan vaadi erillisiä kooditiedostojen latauksia palvelimelle. Tämä vähentää kehitystyön tarvitsemia resursseja.

Kehitetyssä metodologiassa websovellusta kehitetään suoraan samanlaiseen webhotelliin, jossa sovelluksen tuotantoversiota on tarkoitus käyttää. Tämä poikkeaa mainitusta tutkimuksesta ensinnäkin siten, että siinä ei ole kyse webhotellipalveluista vaan etäpalvelimista sinänsä. Lisäksi koodin editointiin päätettiin käyttää verkkopohjaisten koodieditorien sijaan webhotellin sisältämään cPanel- hallintapaneeliin kuuluvaa File Manager-toimintoa, joka mahdollistaa perustoiminnot koodin editointiin. Palvelinkoodin virheenetsintätapakin poikkeaa kyseisestä tutkimuksesta, sillä tässä työssä käytettiin virheenetsintään hallintapaneelin tarjoamaa mahdollisuutta tarkkailla tietokantojen datamuutoksia sovellusten toimintoja kokeiltaessa, tulostamalla palvelinsovelluksen palauttamaa dataa selaimen konsoliin ja näytölle sekä myös diplomityöntekijän tietokoneelle luotavan lokaalipalvelimen avulla. Selainkoodin virheenetsinnässä käytettiin vastaavaa konsoliin ja näytölle tulostusta. Käytetyt menetelmät olivat myös osa manuaalisesti suoritettua testausta, johon kuului myös käyttöliittymän reagoinnin havainnointi eri testitapauksissa. Kehitetyssä metodologiassa kooditiedostojen uudelleenlatauksia palvelimelle tarvittiin vain, kun kehitystyötä tehtiin lokaalipalvelinta käyttäen. Tämä liittyi tyypillisesti sen tyyppisten virheiden etsintään, jota ei voitu toteuttaa muilla edellä kuvatuista tavoista. Edellä mainittu kuvaus on vastaus tutkimuskysymykseen TK1.

Työnhallintajärjestelmän toteuttamiseksi toimeksiantajalta saatiin suuri osa vaatimuksista jo projektin alussa. Kirjallisuuskatsauksen keskeisimmäksi hakutulokseksi itse järjestelmän kannalta valikoitui tutkimus [7], jossa toteutetaan toiminnanohjausjärjestelmän SPA-mallia hyödyntävä prototyyppi. Vaikka toimeksiantaja päätyi jo sovitun projektiaika-

taulun puitteissa vapaata lähdekoodia edustavan, modulaarisen valmisohjelmiston puoleen, työnhallintajärjestelmä kehitettiin silti valmiiksi. Sovitun aikataulun lopussa saatujen lisävaatimusten toteutus joka tapauksessa aiheutti projektin aikataulun venymisen alkuperäisestä.

Lopullinen versio työnhallintajärjestelmästä koostuu asiakasrekisteristä hakukenttineen ja muokkaustoimintoineen, toimenpiteiden lisäystoiminnosta toimenpiteen perustiedot työntekijälle lähettävine tekstiviestitoimintoineen, toimenpiteiden selaus- ja muokkausosion hakukenttineen sekä kalenterin. Kalenteriin toteutettiin myös pääsy toimenpiteiden muokkausosioon valitun toimenpiteen muokkaamiseksi. Sovellus sisältää myös autentikoinnin. Tämä on vastaus tutkimuskysymykseen TK2.

Toteutuksen toiminnot poikkeavat kirjallisuuskatsauksen tuloksena löydetyssä tutkimuksessa [7] toteutetusta toiminnanohjausjärjestelmän SPA-mallia hyödyntävästä prototyypistä ensinnäkin varastonhallintamoduulin sekä taloushallinnosta ja laskutuksesta vastaavan moduulin osalta; kyseiset toiminnot puuttuvat toteutetusta työnhallintajärjestelmästä. Juuri varastonhallinta ja laskutus asetettiin toimeksiantajan taholta järjestelmän jatkokehitystarpeiksi, jotka oli tarkoitus toteuttaa sovitun projektin jälkeen. Myös muut prototyypin moduulit poikkeavat toteutetun järjestelmän toiminnoista, sillä prototyyppi on suunniteltu tukku- ja vähittäiskauppaa harjoittaville yrityksille toimeksiantajan toimialan liittyessä jätevesijärjestelmien huoltoihin ja korjauksiin.

Toteutus poikkeaa mainitusta prototyypistä myös siltä osin, että siinä ei toteutettu Model-View-ViewModel-mallia. Varsinaisesta itsearvioinnista erillisenä johtopäätöksenä todettiin, että seikka vaikuttaa ainakin koodin ylläpidettävyyteen ja selkeyteen; lähes koko sovelluslogiikan sisällyttäminen yhteen selainpään tiedostoon jQuery-kirjaston sisältämän ready-funktion sisään on ratkaisu, jota ei voi pitää selkeänä eikä helposti ylläpidettävänä verrattuna esimerkiksi Model-View-ViewModel-mallilla toteutettuun perusrakenteeseen.

Lisäksi toteutuksessa ei käytetty sovelluskehyksiä kuten tutkimuksen prototyypissä käytetty AngularJS. Myös tähän seikkaan liittyvään parannusehdotukseen havahduttiin työn ollessa valmis myös varsinaisen itsearvioinnin osalta; kyseistä sovelluskehystä käyttämällä olisi voitu lisätä paitsi koodin ylläpidettävyyttä ja selkeyttä, myös testattavuutta. Model-View-ViewModel-mallin tai vastaavan käyttö yhdessä sovelluskehyksen kuten AngularJS:n kanssa nähtiin loppujen lopuksi mahdollistavan ylipäänsä nopeamman ja sujuvamman sovelluskehityksen. Tässä ja edellisessä kappaleessa käsiteltiin myös varsinaisen itsearvioinnin jälkeen todettuja projektiin liittyviä heikkouksia, jotka liittyvät tutkimuskysymykseen TK3. Loput vastaukset tutkimuskysymykseen TK3 saadaan viimeistä lukuun ottamatta seuraavissa kappaleissa.

Myös varsinainen itsearviointi paljasti mahdollisia syitä projektin aikataulun venymiselle. Kehitysprosessien arviointikriteerinä käytettiin pienorganisaatioiden tuottamien tuotteiden laadun ja prosessien tehokkuuden parantamiseen tarkoitetun standardin ISO/IEC 29110 [12] määrittelemiä prosesseja. Kyseiset prosessit koostuvat aktiviteeteista, jotka taas koostuvat tehtävistä. Toteutettua ohjelmistoprojektia verrattiin kyseisiin tehtäviin sen osalta kuin se yhden henkilön projektissa oli tarkoituksenmukaista. Vertailun tuloksena prosessien heikoimmat aktiviteetit koottiin yhteen niiden valintaan johtaneine tehtävineen ja asetettiin ne järjestykseen niiden keskinäisten juurisyy-yhteyksien mukaan niin, että kaikkien heikkouksien juurisyyaktiviteetti on ensimmäisenä. Tämän tuloksena heikkouksien juurisyy on ohjelmistokomponenttien luomiseksi vaadittavien tehtävien puuttuminen. Sen seurauksena tarkoituksenmukaisia ohjelmistokomponentteja ei tunnistettu, suunniteltu eikä niitä dokumentoitu standardin määrittelemällä tavalla. Näin ollen, sovellusta ei myöskään rakennettu erillisinä komponentteina, niiden testitapauksia ei tehty standardin määrittelemällä tavalla eikä selkeän komponenttirakenteen puuttuessa myöskään ohjelmiston kokoonpanoa tehty standardin mukaisesti. Edellä mainituista syistä sovelluslogiikka tulkittiin monimutkaistuneen kehitystyön edetessä niin, että sekä itse kehitystyö, että siihen kuuluvat testaukset kuluttivat tarpeettoman paljon aikaa. Lisäksi käytetty manuaalinen testaustapa arvioitiin vieneen odotettua enemmän aikaa.

Koska suurin osa sovelluksen toiminnoista on toimeksiantajan ideoimia, itse sovelluksen toimintoja ja ominaisuuksia arvioitiin vain käyttöönottovaihetta edeltävän testauksen perustella. Kyseisen testauksen perusteella sovellus täyttää toimeksiantajan sille asettamat vaatimukset. Toimeksiantajan vetäydyttyä projektista, sovelluksen lopullista versiota ei hyväksytetty, vaan sovelluksen arviointi jäi mainitun testauksen varaan. Lopullisen version hyväksytys toimeksiantajalla olisi tuonut olennaista lisäarvoa sovelluksen arviointiin.

Toimeksiantaja esitti projektin alussa järjestelmälle alustavia laajennustarpeita, jotka oli tarkoitus toteuttaa sovitun projektin jälkeen. Laajennusosia olisivat esitetyn mukaan olleet ainakin varastonhallinta ja laskutus. Mainitusta syystä johtuen, kehitetyn sovelluksen itsearviointiin päätettiin sisällyttää myös sovelluksen laajennettavuus juuri kyseisten laajennusosien kannalta. Valmiiden, vapaata lähdekoodia edustavien varastonhallinta- ja laskutusmoduulien saatavuuden toteutetun kaltaiseen sovellukseen todettiin olevan kriittinen tekijä laajennettavuuden kannalta. Niiden saatavuuteen liittyvä selvitystyö ei tosin kuulunut tämän työn aihepiiriin. Itse sovelluksen keskeisimpien entiteettien; asiakas- ja toimenpideobjektien käyttö globaalien objektilistojen kautta todettiin kuitenkin laajennettavuutta edistäväksi tekijäksi. Toisaalta ohjelmistokomponenttien luomiseksi vaadittavien tehtävien puuttuminen nähtiin juurisyyksi myös sille, että sovelluksen laajennetta-

vuus ei ollut paras mahdollinen. Tekijän nähtiin olevan yhteydessä myös toteutetun koodin heikohkoon luettavuuteen. Jako komponenteiksi nähtiin olevan mahdollista myös ilman sovelluskehyksiä JavaScriptin oliosyntaksin avulla. Jonkin sovelluskehyksen käyttö yhdessä Model-View-ViewModel-mallin tai vastaavan kanssa on kuitenkin tämän työn perusteella suositeltavaa.

Työssä käytetyistä työkaluista arvioitiin webhotellin cPanel-hallintapaneeliin kuuluvan koodieditointitoiminnon sekä lokaalipalvelimen käytettävyys virheenetsinnässä käytetyn metodologian yhteydessä. Mainittu tapa editoida koodia nähtiin käytettävyydeltään haasteelliseksi varsinkin suurten kooditiedostojen tapauksissa, joiden yhteydessä kaivattiin hakutoimintoa, jonka avulla etsittävä osa koodista olisi löydetty nopeasti. Toisaalta edellä kuvattu suositus kehitettävän koodin jaosta komponenteiksi yhdessä tarkoituksenmukaisen sovelluskehyksen kanssa mahdollistaa myös lyhyempien kooditiedostojen käytön, jolloin hakutoiminnon tärkeys on vähäisempi. Tällöin kyseisen koodieditointitoiminnon ominaisuudet riittänevät työssä käytetyn kehittämismetodologian osaksi, ellei muuta editointimahdollisuutta ole käytettävissä. Lokaalipalvelimen käyttöön todettiin liittyvän samankaltaisia resurssitarpeita kuin kirjallisuuskatsauksen tuloksena löydetyn tutkimuksen [17] lähtökohdissa; koodin synkronointi kehittäjän tietokoneen ja etäpalvelimen välillä. Tutkimuksen esittämässä metodologiassa ei mainittua synkronointitarvetta ole, sillä palvelinkoodin virheenetsintään käytetään vain verkossa toimivia ohjelmointiympäristöjä. Työssä tehtiinkin johtopäätös, jonka mukaan kyseisen ratkaisun soveltuvuutta myös webhotellikäyttöön kannattaa tutkia. Tämä onnistuu mahdollisesti SSH-yhteyden avulla. Tähän pääteltiin liittyvän mahdollisuus myös verkossa toimivien koodieditorien käyttöön. Tällöin vältyttäisiin esim. tässä työssä käytetyn File Manager-toiminnon vaatimattomien editointiominaisuuksien tuomilta haasteilta.

Työssä käytettyä metodologiaa tässä luvussa esitettyine jatkokehityssuosituksineen voinee hyödyntää vastaavissa etänä toteutettavissa websovellusprojekteissa, jotka perustuvat inkrementaalisen kehitystavan käyttöön. Metodologian vahvuus on kehitysvaiheiden tuotantoympäristöä vastaava visualisointimahdollisuus suoraan webhotelliin kehitettävän sovelluksen kautta. Käyttämällä kehitystyössä samanlaista webhotellia kuin jossa tuotantoversiota on tarkoitus käyttää mahdollistaa suoraviivaisemman käyttöönoton. Tässä työssä työnhallintajärjestelmät luokiteltiin toiminnanohjausjärjestelmien osajoukoksi. Työssä tehtiin myös johtopäätös, jonka mukaan niiden kehittäminen alusta asti itse niitä tarvitsevien yritysten toimesta on harvinaista. Itse kehitettynä valmiiden vapaata lähdekoodia edustavien moduulien saatavuus ja integroitavuus noussee ratkaisevan tärkeäksi.

# LÄHTEET


[1] R. Agarwal, D. Umphress, Extreme programming for a single person team. In: Proc. of the 46th Annual Southeast Regional Conference on XX (2008), 82–87.

[2] AngularJS verkkosivusto. Saatavissa (viitattu 8.4.2021): https://angularjs.org/

[3] C. Dorman, V. Rajlich, Software change in the solo iterative process: an experience report. In: Proc. of Agile Conference (2012), 21–30.

[4] W.W. Eckerson Three tier client/server architectures: achieving scalability, performance, and efficiency in client/server applications 1995. Open Information Systems 3(20) 46-50

[5] H.-C. Estler, M. Nordio, C. A. Furia, and B. Meyer, "Collabotative debugging," IEEE 8th Int. Conf. on Global Software Engineering, pp. 110–119, 2013.

[6] E. A. Ganin, "An approach to the development of scalable web applications," Prospects for the development of information technology, no. 12, pp. 62–66, 2013.

[7] Gunawan J., Kosala R.R., Genie Enterprise Resource Planning for Small Medium Enterprises Implementing Single Page Web Application, IOP Conference Series: Earth and Environmental Science 426(1),012170 2020. Saatavissa (viitattu 6.1.2021): https://www-scopus-com.libproxy.tuni.fi/search/form.uri?display=basic

[8] I. Haikala, T. Mikkonen, Ohjelmistotuotannon käytännöt, 12. Painos, Talentum Media Oy, Helsinki, 2011, 242 s.

[9] W3 Schools, JSON – Introduction. Verkkosivu. Saatavissa (viitattu 8.4.2021): https://www.w3schools.com/js/js_json_intro.asp

[10] W. S. Humphrey: The Personal Software Process (PSP). Technical report, Carnegie Mellon University, 2000.

[11] ISO/IEC 33020:2019 Information technology — Process assessment — Process measurement framework for assessment of process capability. Saatavissa (viitattu 16.2.2021): https://www.iso.org/standard/78526.html

[12] ISO/IEC TR 29110-5-1-1:2012 Software engineering — Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-1: Management and engineering guide: Generic profile group: Entry profile. Saatavissa: https://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[13] T. Karttunen, Utilization of prototyping methods in user-centered design process, Tampereen teknillinen yliopisto, 2017, 87 s.

[14] M. Kähkönen, Itsenäisen ohjelmistokehittäjän ketterät menetelmät, pro gradu-tutkielma, Tampereen yliopisto, 2014, 42 s.

[15] T. V. Lagutkina, “A review of architectural decisions of version control systems,” Prospects for the Development of Information Technologies, no. 24, pp. 18–22, 2015.

[16] R. G. Lyubimtsev, “Eclipe environment for the development of java language programs,” Information and communication technologies in teacher education, no. 2(30), pp. 40–42, 2014.

[17] A. K. Myo, E.M. Portnov, S.V. Tsymbalov, Method for Increasing High Speed of Development of Remote Web Application, International Russian Automation Conference (RusAutoCon), IEEE 2020. Saatavissa (viitattu 11.1.2021): https://ieeexplore-ieee-org.libproxy.tuni.fi/document/9208029

[18] A., Nyström, Defining and Evaluating an Agile Software Development Process for a Single Software Developer. Master's thesis, Chalmers University of Technology, 2011.

[19] A. A. Pupykina, “Analysis of modern approaches to the automation of web application development,” Intersectoral information service, no. 3, pp. 12–20, 2011.

[20] Rajapinnat ja REST, Web-palvelinohjelmointi Java 2019, Mooc.fi 2019. Saatavissa (viitattu 8.4.2021): https://web-palvelinohjelmointi-19.mooc.fi/osa-6/4-rajapinnat-ja-rest

[21] V. Rajlich, Software Engineering: The Current Practice. Chapman and Hall, 2011.

[22] G. Shanks, P. B. Seddon, L.P.Willcocks, Introduction—ERP – The Quiet Revolution?, Cambridge University Press 2003. Saatavissa: https://library.books24x7.com/

[23] A. V. Sorokin and D. V. Koznov, “Overview ECLIPSE modeling project,” System Programming, vol. 5, no. 1, pp. 6–32, 2010.

[24] M. A. Sultanov, A. M. Marasulov, and Z. E. Ibraeva, “Analysis of approaches to the development of web applications,” Innovative information technologies, vol. 1, no. 2, pp. 410–412, 2013.

[25] Tilastokeskus 2017, Liiketoiminnan sähköistyminen. Saatavissa (viitattu 27.3.2020): https://www.stat.fi/til/icte/2017/icte_2017_2017-11-30_kat_005_fi.html

[26] Two-way binding, Angular-verkkosivu. Saatavissa (viitattu 8.4.2021): https://angular.io/guide/two-way-binding

[27] Web-palvelimen toiminta, Web-palvelinohjelmointi Java 2019, Mooc.fi 2019. Saatavissa (viitattu 8.4.2021): https://web-palvelinohjelmointi-19.mooc.fi/osa-1/2-web-palvelimen-toiminta

[28] W3 Schools, What is npm? Verkkosivu. Saatavissa (viitattu 8.4.2021): https://www.w3schools.com/whatis/whatis_npm.asp

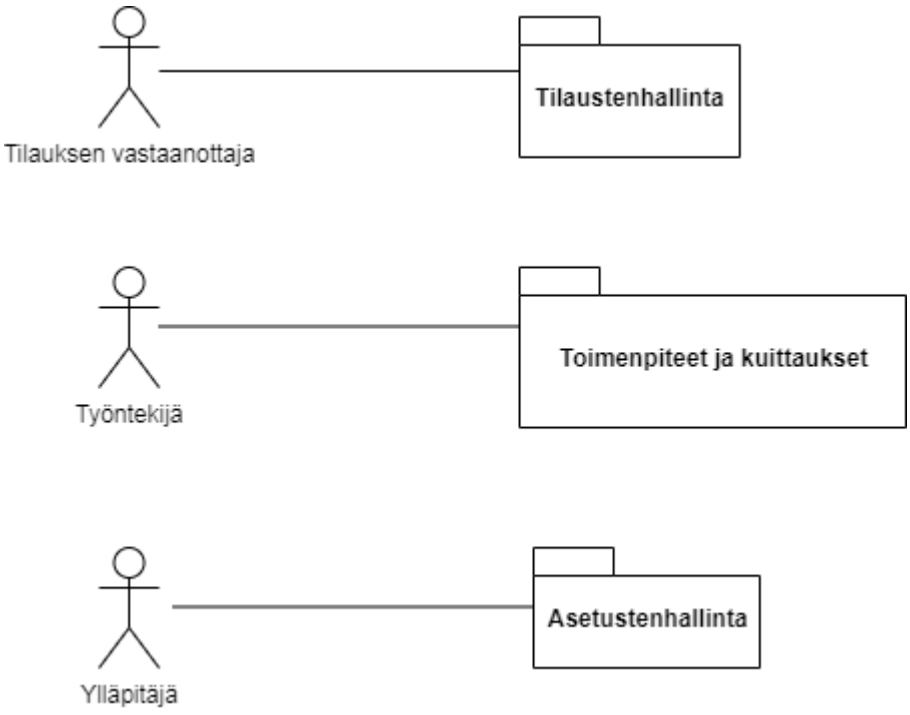
# LIITE A: ASIAKASVAATIMUSTEN KUVAUS

Tässä liitteessä on kuvattu pääosa esitetyistä asiakasvaatimuksista. Vaatimustenkäsittelyn ensimmäisiä tuotoksia oli toimeksiantajan kanssa kasvotusten pidetyssä neuvonpidossa ja myöhemmän kommunikaation kautta määritellyt vaatimukset, jotka on koottu alla esitettyyn listaan. Sen lisäksi vaatimuksia annettiin vielä lisää varsinkin projektin viimeisessä tapaamisessa. Listassa esitettyihin lyhenteisiin viitataan liitteen myöhemmissä osissa ja liitteessä B, jossa kuvattava toiminnallinen määrittely on luotu koko projektin aikana saaduista vaatimuksista.

- VA1: Työnhallintajärjestelmä on websovellus, jonka navigointi toteutetaan sivuston ylälaitaan sijoitettavan valikkorivin avulla
- VA2: Valikkorivin tulee sisältää osiot ”asiakasrekisteri”, ”luo toimenpide”, ”selaa toimenpiteitä”, ”kalenteri” ja ”asetukset”
- VA3: Asiakasrekisterinosion täytyy listata asiakkaat
- VA4: Asiakasrekisteriosion täytyy sisältää lomake, jolla asiakasrekisteriin voi lisätä uusia asiakkaita
- VA5: Asiakasrekisteriosion täytyy sisältää lomake asiakastietojen muokkausta varten.
- VA6: Luo toimenpide - osion täytyy sisältää lomake, jolla tietylle asiakkaalle luodaan uusia toimenpiteitä.
- VA7: Eri toimenpidetyyppejä ovat huoltokäynti, vuosihuoltokäynti, vuosihuoltosopimuskäynti, lähetä tuote ja soittopyyntö. Toimenpiteen valinta tulee olla mahdollista uutta toimenpidettä luotaessa ja kukin toimenpide tulee näkyä kalenterissa uniikilla taustavärillä.
- VA8: Luo toimenpide - osion tulee huomioida se, onko asiakas lisätty jo aiemmin järjestelmään niin, että jos kyseessä on uusi asiakas, järjestelmä lisää sen automaattisesti asiakasrekisteriin.
- VA9: Luo toimenpide – osion lomakkeen tuli sisältää automaattisen täyttötoiminnon niin, että jos asiakas on jo lisätty järjestelmään ja lomakkeessa olevaan asiakkaan nimelle tarkoitettuun tekstikenttään aletaan kirjoittaa, järjestelmä etsii aiemmin lisättyjen asiakkaiden nimistä tekstikentän syötettä ja tarjoaa mahdollisuuden täyttää lomake asiakasrekisteristä löytyvän osuman tiedoilla.
- VA10: Luo toimenpide – osion lomakkeessa tulee olla myös valintamahdollisuus sille, kenelle yrityksen työntekijöistä toimenpide osoitetaan.
- VA11: Luo toimenpide – osion lomakkeen tallennuksen yhteydessä tulee olla myös erillinen painike sille, että tallennuksen lisäksi toimenpiteen perustiedot lähetetään tekstiviestinä työntekijälle, jolle toimenpide osoitetaan
- VA12: Selaa toimenpiteitä – osion tulee listata kaikki toimenpiteet

- VA13: Selaa toimenpiteitä – osion tulee sisältää muokkaustoiminto vastaavanlaista lomaketta käyttäen kuin luo toimenpide – osiossa.
- VA14: Kalenterin tulee sijoittaa lisätyt toimenpiteet automaattisesti toimenpiteelle asetetun päivämäärän ja kellonajan kohdalle.
- VA15: Työnhallintajärjestelmän on toimittava sekä työpöytä- että mobiililaitteilla.
- VA16: Työnhallintajärjestelmässä on autentikointi niin, että vain yrityksen työntekijät pääsevät käyttämään sitä.

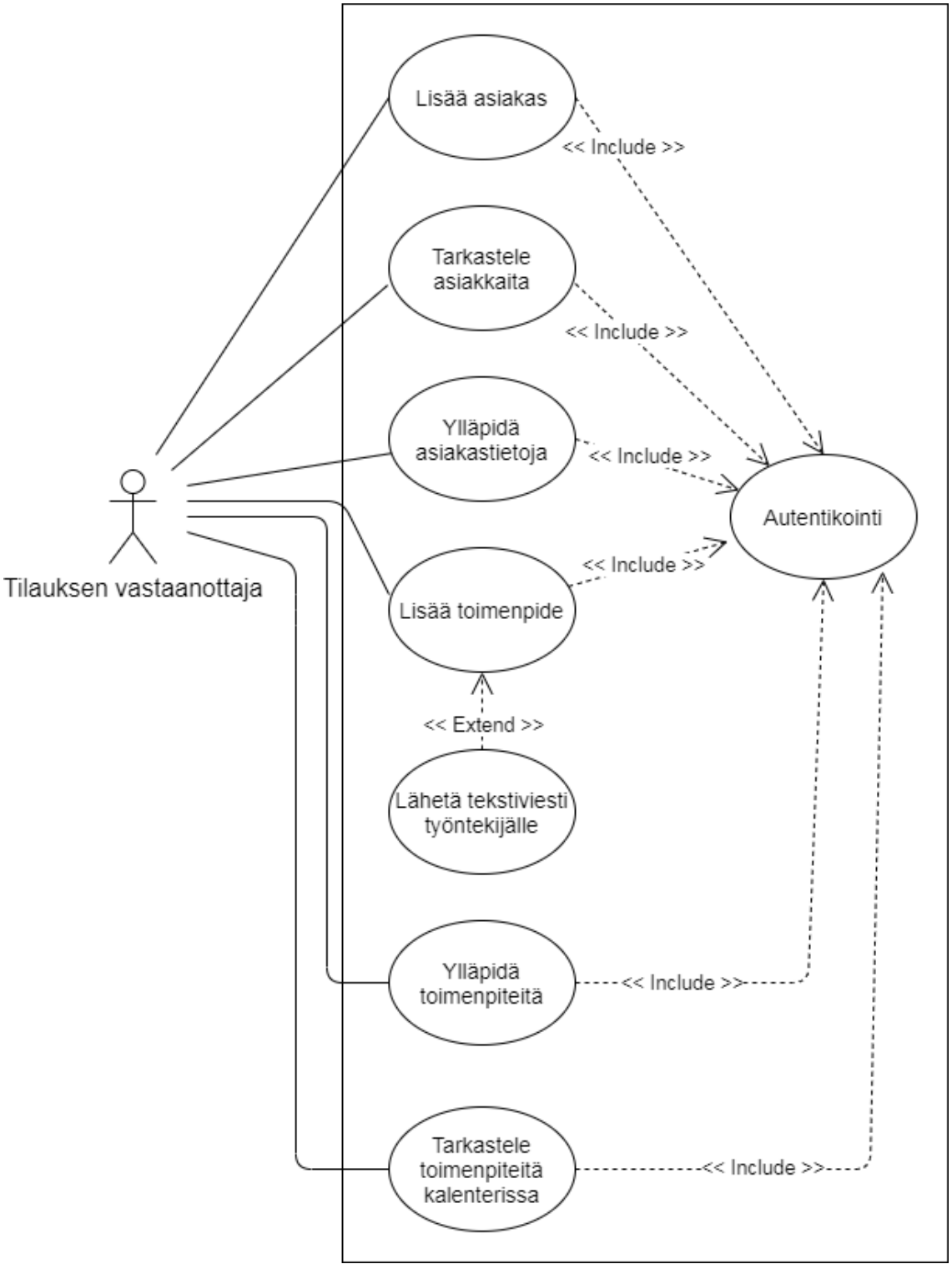
Asiakasvaatimukset koottiin kolmeen pakettiin sen mukaan kuin systeemin kanssa vuorovaikutuksessa olevia käyttäjärooleja tunnistettiin (kuva 1). Tilauksen vastaanottaja katsottiin vastaavan tilaustenhallinnasta, työntekijälle katsottiin olevan tärkeää päästä käsiksi toimenpidetietoihin sekä kuittaamaan niitä valmiiksi. Ylläpitäjälle oli luontevaa antaa vastuu asetustenhallinnasta.




***Kuva 1. Asiakasvaatimusten paketointi käyttäjäroolien perusteella.***

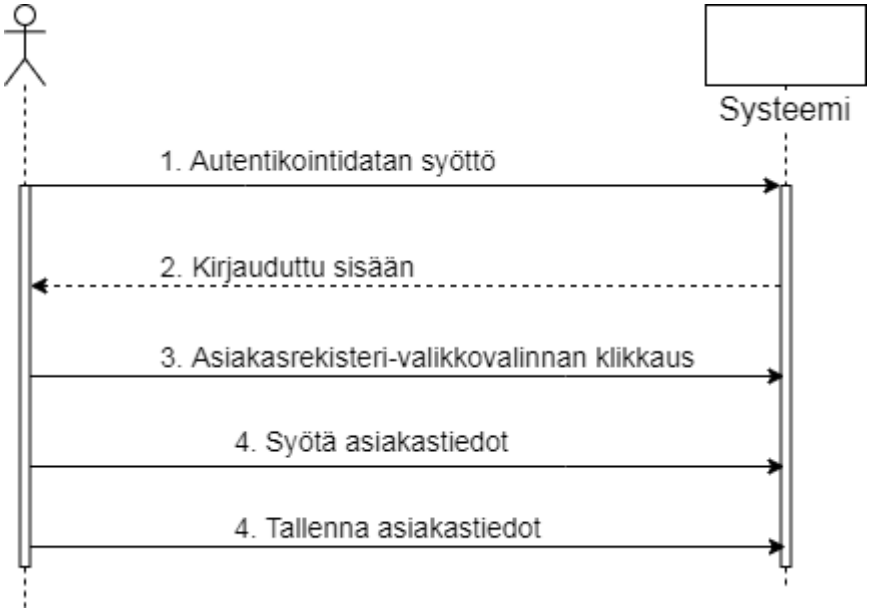
Tilauksen vastaanottajan vastuulla katsottiin olevan kuvan 2 mukaiset käyttötapaukset. Ne ovat asiakasvaatimukset VA3-VA6, VA11-VA14 ja VA16 esitettynä käyttötapauksina käyttötapauskaaviossa. Esimerkiksi eri käyttötapausten yhteyteen include-merkinnällä

lisätty Autentikointi – osa on kuvaus vaatimuksesta VA16. Muut asiakasvaatimukset kuvataan jäljempänä tarkennettujen käyttötapausten yhteydessä.



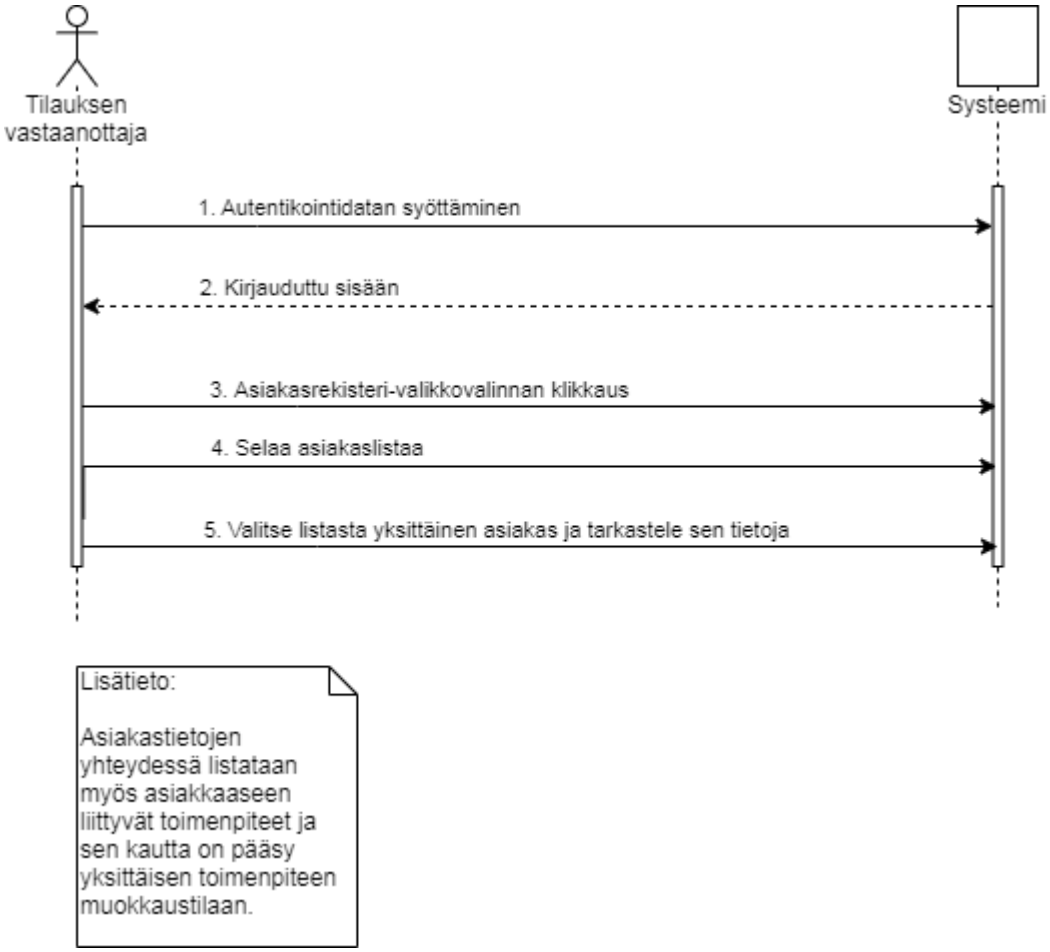

***Kuva 2. Tilauksenhallinta-paketin sisältämät käyttötapaukset.***

Käyttötapauksia kuvattiin tarkemmin sekvenssikaavioiden avulla. Kuva 3 on Lisää asiakas-käyttötapauksen tarkennettu kuvaus ja se kuvaa asiakasvaatimusta VA4.



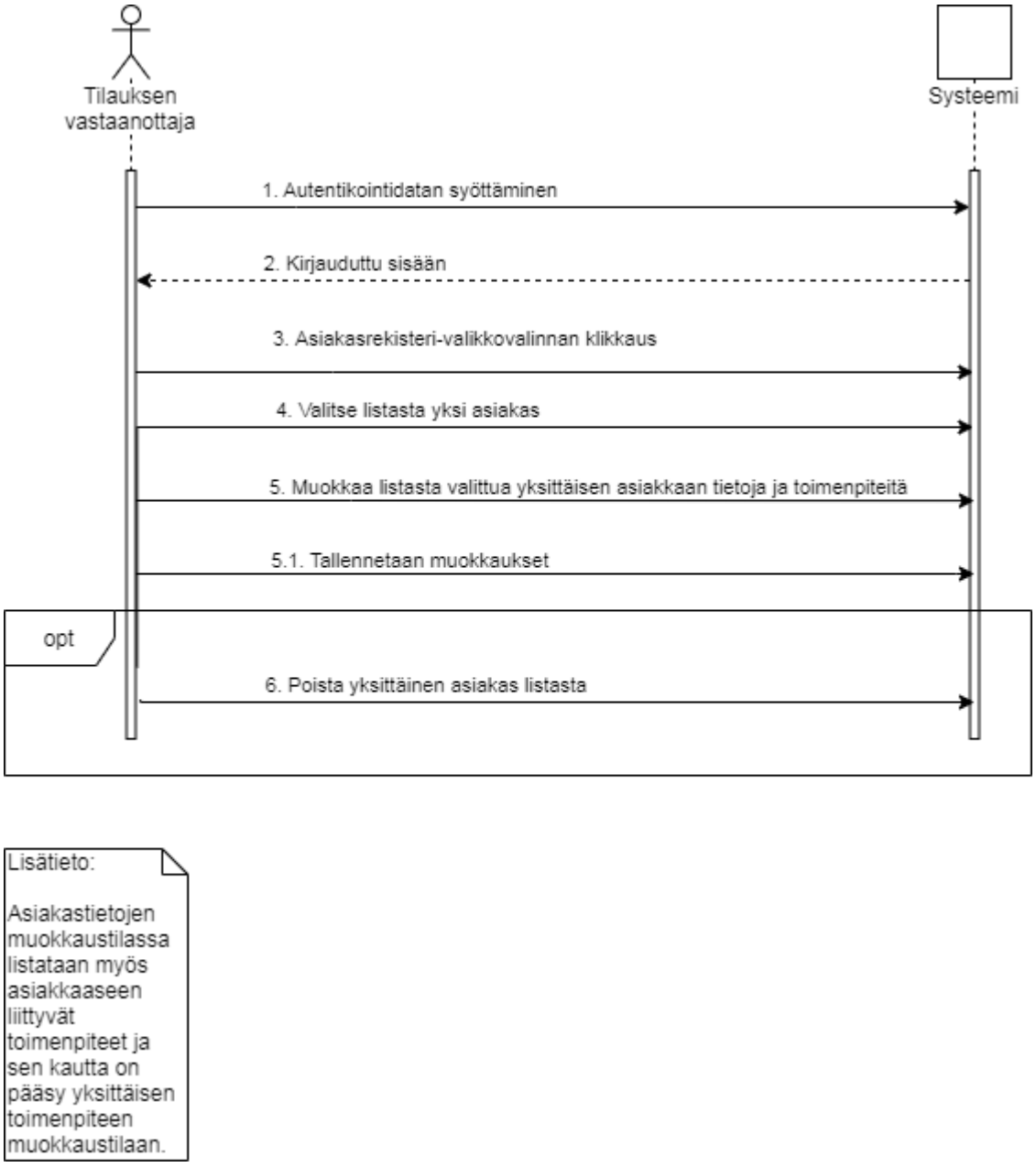

***Kuva 3. Tarkennettu kuvaus asiakasvaatimuksesta VA4 sekvenssikaaviona.***

Tarkastele asiakkaita – käyttötapausta (asiakasvaatimus VA3) varten luotiin kuvan 4 tarkennettu kuvaus käyttötapauksesta. Vaatimukset VA3 ja VA4 liittyvät siis Asiakasrekisteri – nimiseen valikkovalintaan. Asiakastietojen yhteydessä näytettävistä toimenpiteistä ei lisätty erillistä vaatimusta liitteen alussa esitettyyn listaan, vaan se saatiin myöhemmin toimeksiantajalta. Sama vaatimus on esitetty myös kuvan 5 lisätiedoissa.




***Kuva 4. Tarkennettu kuvaus käyttötapaukseen Tarkastele asiakkaita liittyen.***

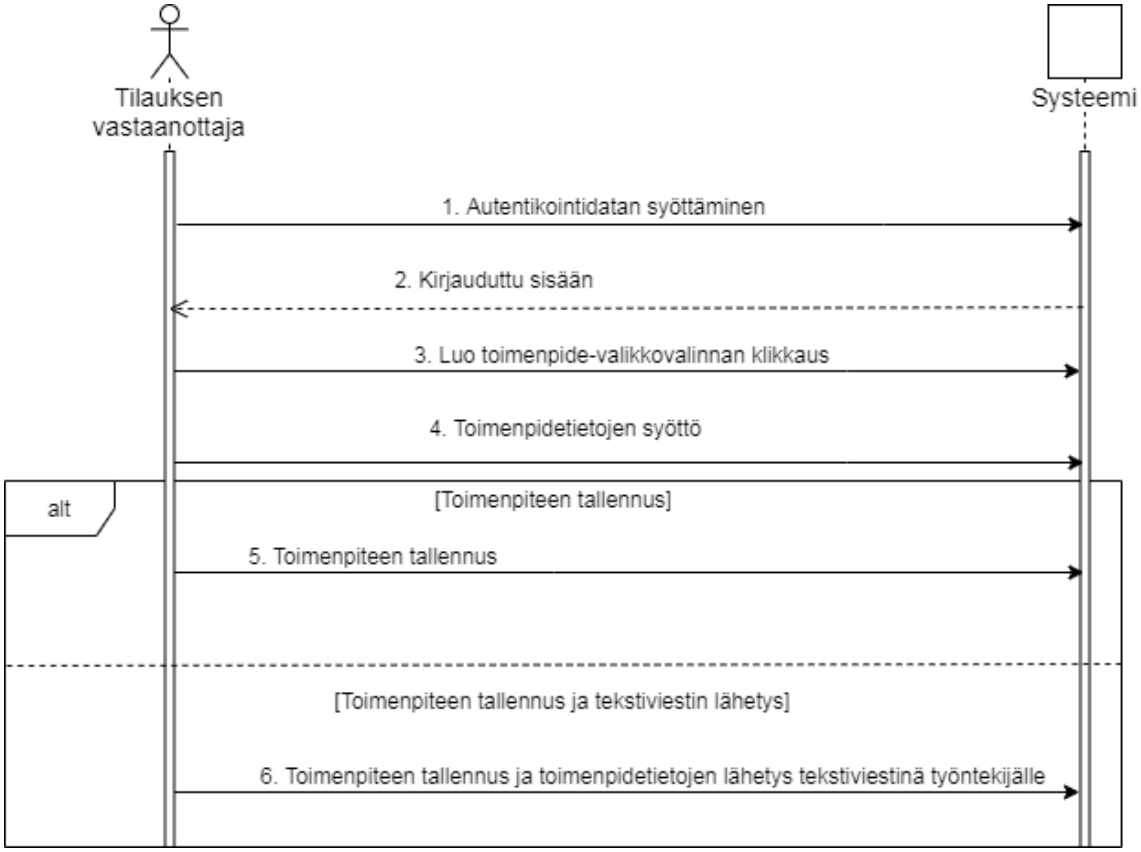
Käyttötapaukseen Ylläpidä asiakastietoja liittyvä tarkennettu kuvaus on kuvassa 5. Tarvetta asiakkaan poistotoiminnosta ei esitetty toimeksiantajalta, vaan se johdettiin itse annettujen vaatimusten perusteella.




***Kuva 5. Ylläpidä asiakastietoja – käyttötapauksen tarkennettu kuvaus sekvenssikaaviona.***

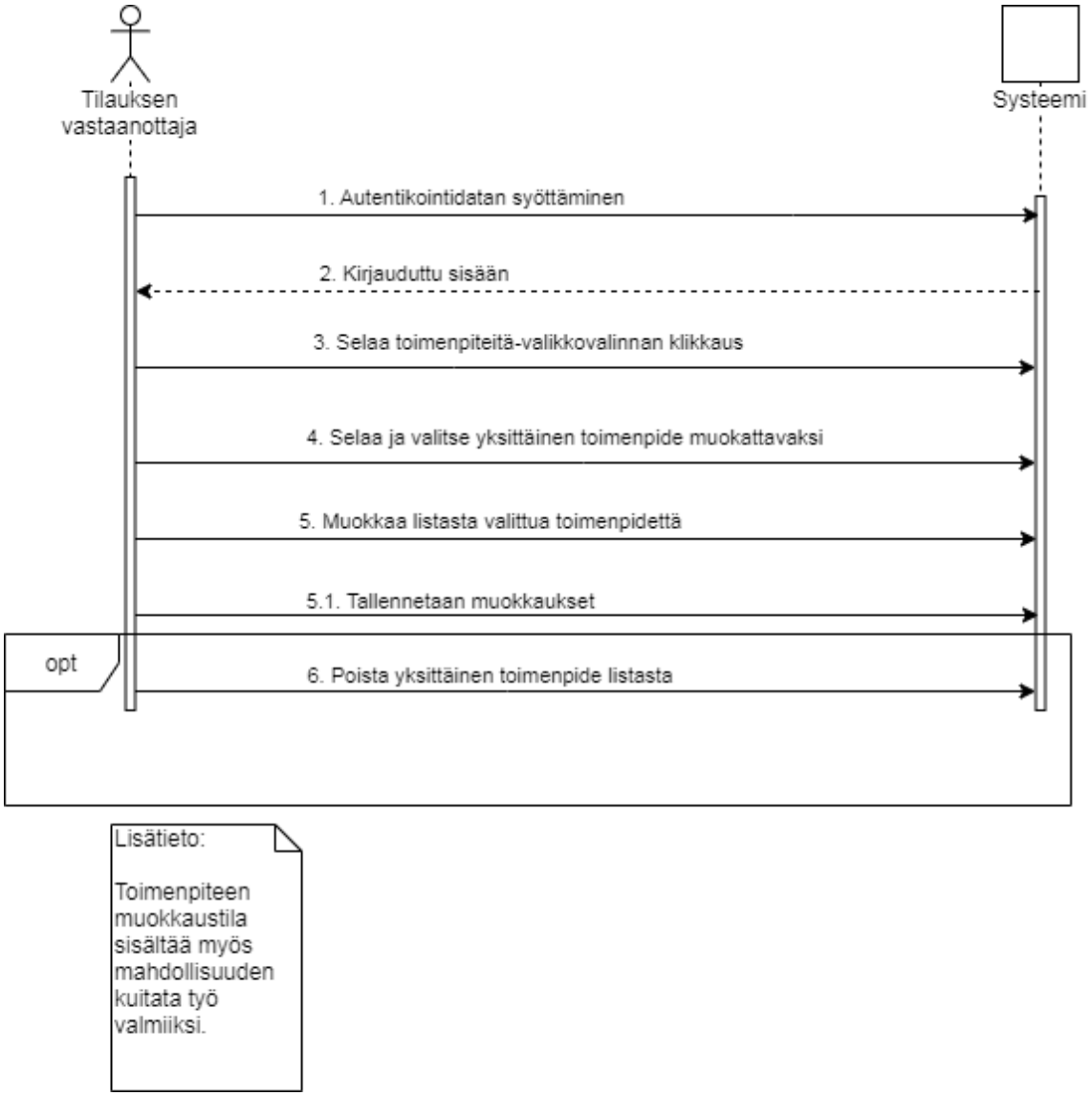
Vaatimusten VA2 ja VA6 mukaisesti työnhallintajärjestelmän webkäyttöliittymässä tulee olla Luo toimenpide-valikkovalinta, joka on kuvattu lähes samannimisenä käyttötapauksena kuvassa 2. Itse mahdollisuus toimenpiteen lisäyksestä liittyy vaatimukseen VA6. Vaatimus VA11 toimenpiteen perustietojen lähettämisestä tekstiviestinä työntekijälle, jolle toimenpide osoitetaan, on kuvattu Luo toimenpide-käyttötapauksen laajennussuhteena (extend) samassa kuvassa. Tarkennettu kuvaus mainitusta käyttötapauksesta (vaatimus VA6) sekvenssikaaviona kuvassa 6. Kaaviossa on käytetty alt-kehystä, jolla

on kuvattu vaihtoehtoja toimenpiteen tallennuksen ja toimenpiteen tallennuksen ja sen perustietojen lähettämisen työntekijälle välillä.



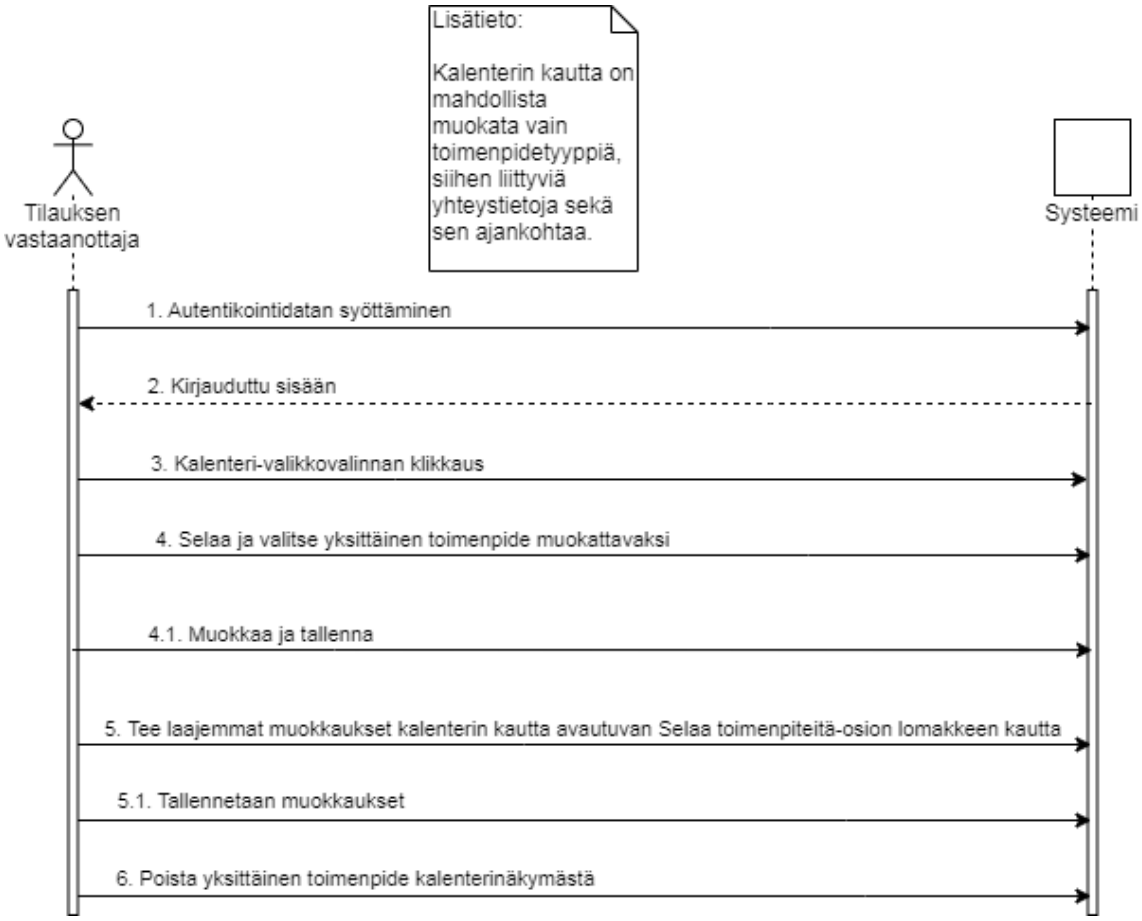

***Kuva 6. Tarkennettu kuvaus käyttötapaukseen Luo toimenpide liittyen.***

Käyttötapaus Ylläpidä toimenpiteitä liittyy ensisijaisesti vaatimukseen Selaa toimenpiteitä – valikkovalinnan ominaisuuteen listata kaikki toimenpiteet (VA12) ja saman osion ominaisuuteen sisältää toimenpiteen muokkaustoiminto samanlaista lomaketta käyttäen kuin Luo toimenpide-osiossa (VA13). Tarkennettu kuvaus mainitusta käyttötapauksessa kuvassa 7.



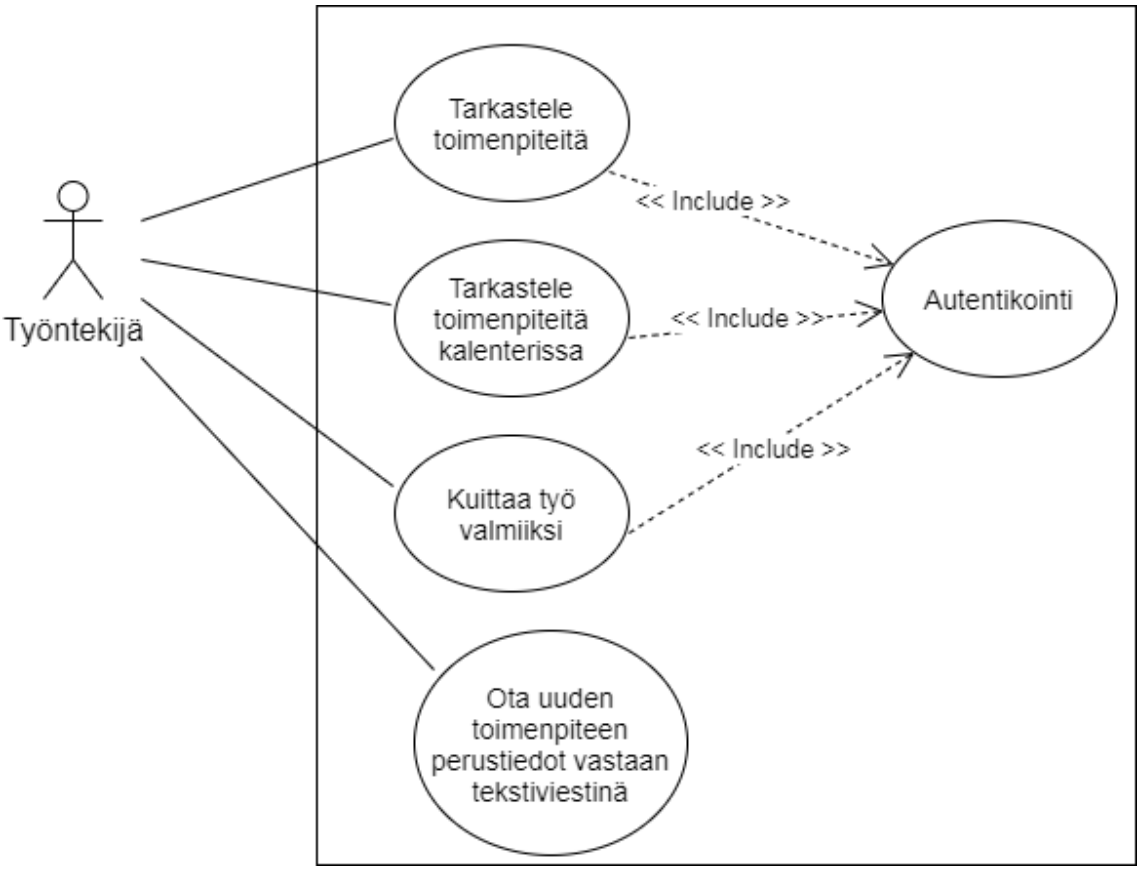

***Kuva 7. Ylläpidä toimenpiteitä – käyttötapauksen tarkennettu kuvaus sekvenssikaaviona.***

Käyttötapaus Tarkastele toimenpiteitä kalenterissa sisältää paitsi kalenteriin ajoitetut toimenpiteet, myös niiden muokkausmahdollisuuden kalenterin yhteydessä sekä linkin Selaa toimenpiteitä – osion muokkauslomakkeelle ko. toimenpiteen tiedot sille valmiiksi generoituna. Kalenterin olemassaolo liittyy vaatimukseen VA2 ja toimenpiteiden ajoitus kalenteriin vaatimukseen VA14. Niiden lisäksi toimeksiantaja esitti kuvassa 8 esitetyt vaatimukset kehitettävälle websovellukselle.




***Kuva 8. Tarkastele toimenpiteitä kalenterissa – käyttötapauksen tarkennettu kuvaus sekvenssikaaviona.***

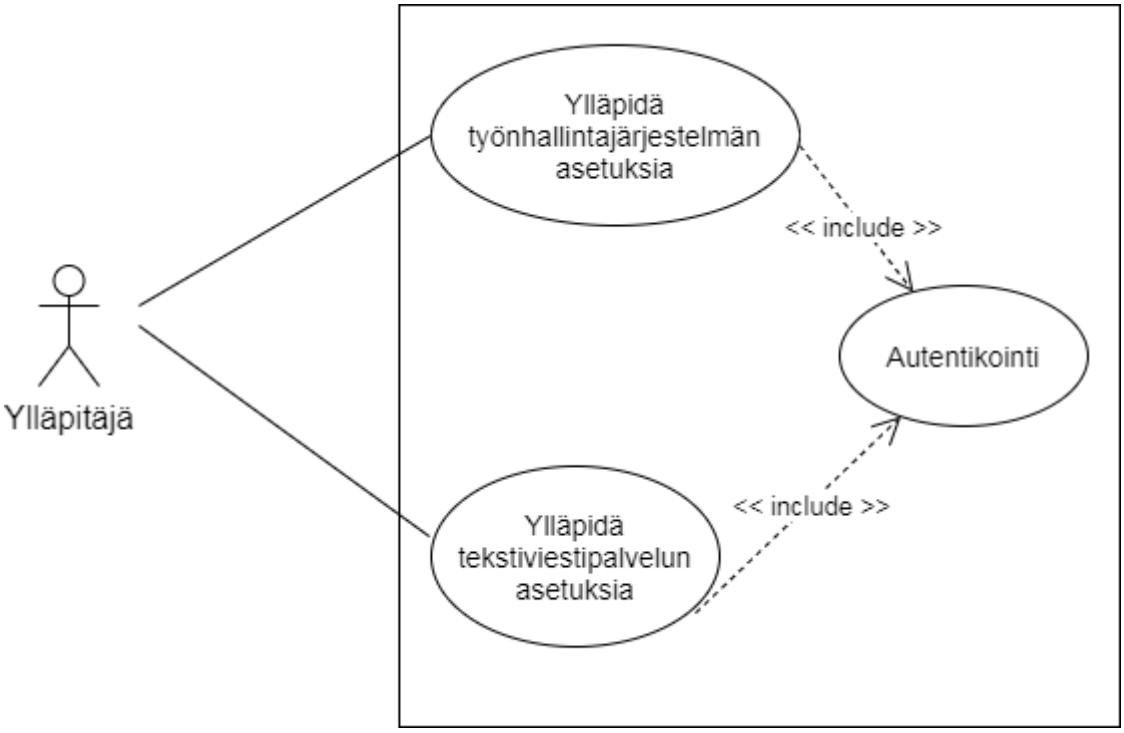
Paketin Toimenpiteet ja kuittaukset käyttötapaukset ovat esitettynä kuvassa 9. Ne kuvaavat osin samoja toimeksiantajan suoraan antamia vaatimuksia kuin paketin Tilauksenhallinta sisältävät käyttötapaukset. Käyttötapaus Tarkastele toimenpiteitä kalenterissa on jopa samanniminen sillä erotuksella, että työntekijä – roolille kuuluu vain toimenpidetietojen tarkastelu kalenterissa – lähinnä niiden ajankohtien osalta. Muilta osin tarkennettu kuvaus kuvassa 8 vastaa tilauksen vastaanottajan vastaavaa käyttötapausta.




***Kuva 9. Toimenpiteet ja kuittaukset - paketin sisältämät käyttötapaukset.***

Tarkastele toimenpiteitä – käyttötapaus on luotu asiakasvaatimuksien VA2 ja VA12 perusteella ja tarkennettu kuvaus kuvassa 7 on kuvaus myös tästä käyttötapauksesta kohtaan 5 asti sillä erotuksella, että tässä vain tarkastellaan lomakkeelle generoituja toimenpidetietoja. Myös kuittaa työ valmiiksi – käyttötapaus on kuvattu samassa sekvenssikaaviossa, sillä asiakasvaatimukseen kuuluu, että työ on voitava kuitata valmiiksi Selaa toimenpide – osion yksittäisen toimenpiteen muokkaukseen tarkoitetulla lomakkeella. Käyttötapaus Ota uuden toimenpiteen perustiedot vastaan tekstiviestinä liittyy kuvan 6 tarkennettuun kuvaukseen käyttötapauksesta Luo toimenpide ja myös siis ensin mainittu käyttötapaus liittyy vaatimukseen VA11.

Asiakasvaatimus VA2 liittyy Asetukset-nimiseen käyttöliittymän valikkovalintaan. Koska vaatimuksen VA11 mukaan kehitettävässä työnhallintajärjestelmässä tulee olla mahdollisuus lähettää tekstiviesti työntekijälle uuden toimenpiteen luonnin yhteydessä, pääteltiin, että mainitussa osiossa täytyy olla mahdollista ylläpitää tekstiviestipalvelun asetuksia. Vaatimuksen VA16 perusteella taas pääteltiin, että autentikointi toteutetaan käyttäjätunnusta ja salasanaa käyttäen, joita tulee olla mahdollista vaihtaa kyseisen osion kautta. Kuvassa 12 on esitetty pääteltyihin vaatimuksiin liittyvät käyttötapaukset.




***Kuva 10. Asetuksenhallinta – paketin käyttötapaukset.***

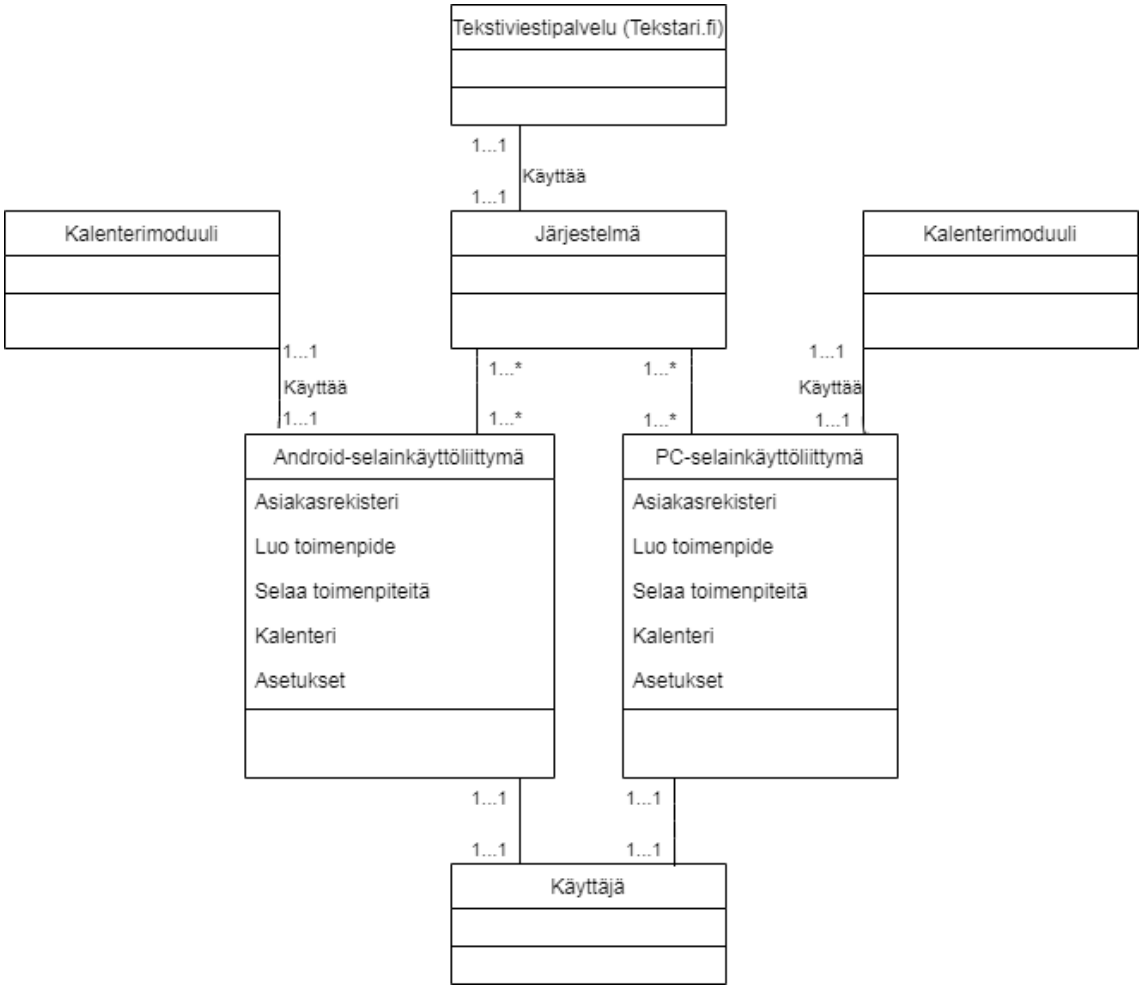
# LIITE B: TOIMINNALLINEN MÄÄRITTELY

Tässä liitteessä ohjelmistovaatimuksia käsitellään olennaisimmiksi katsottujen kohtien osalta ja siltä osin kuin niitä ylipäänsä johdettiin asiakasvaatimuksista. Ulkoisten liittymien osalta kuvauksessa käytetään luokkakaavion avulla toteutettua liittymäkaaviota, toimintojen mallinnuksessa tilakaavioita ja tietokannan mallinnuksessa luokkakaaviota. Koska toiminnallinen määrittely on kokonaisuus, joka perustuu asiakasvaatimuksista johdettuihin ohjelmistovaatimuksiin, tässä liitteessä käytetyt kaaviot kuvaavat myös asiakasvaatimuksia.

- VA1 ⟹ OV1: Työnhallintajärjestelmän toimittava selaimessa
- VA1 ⟹ OV2: Työnhallintajärjestelmä toimii webhotellipalveluntarjoajan palvelimella.
- VA2 ⟹ OV3: Sovelluksessa on jonkinlainen kalenteritoiminto
- VA7 ⟹ OV4: Lomakkeessa tulee olla mahdollisuus valita yksi toimenpide vaatimuksessa VA7 esitetystä joukosta
- VA8 ⟹ OV5: Toimenpiteen tallennuksen yhteydessä ohjelma tarkistaa ensin, onko asiakas jo rekisterissä ja jos ei ole, lisätään uusi asiakas ja uusi toimenpide tallennetaan niin, että se sisältää uuden asiakkaan uniikin tunnisteen, muussa tapauksessa tallennetaan toimenpide valittuun asiakkaaseen liittyvällä uniikilla tunnisteella varustettuna. Viimeksi mainitussa tapauksessa päivitetään myös valitun asiakkaan tiedot lomakkeeseen syötettyjen tietojen mukaan.
- VA9 ⟹ OV6: Toiminnon tulee näyttää syötekentän tuntumassa syötteen perusteella generoitu lista, josta on mahdollista valita haluamansa nimi.
- VA11 ⟹ OV7: Ohjelma lähettää toimenpiteen perustiedot tekstiviestipalvelun tarjoajalle
- VA3, VA12 ⟹ OV8: Valikkovalintaan liittyvä lista generoidaan aina, kun osioon johtavaa navigointivalintaa klikataan
- VA13 ⟹ OV9: Muokkaukseen käytettävä lomake täytetään asiakkaan tiedoilla, jolle toimenpide on osoitettu sekä itse toimenpiteen tiedoilla muokkaustilaan siirryttäessä.
- VA14 ⟹ OV10: Toimenpiteelle on asetettava alustava aloitus ja lopetusaika uuden toimenpiteen tallennuksen yhteydessä.

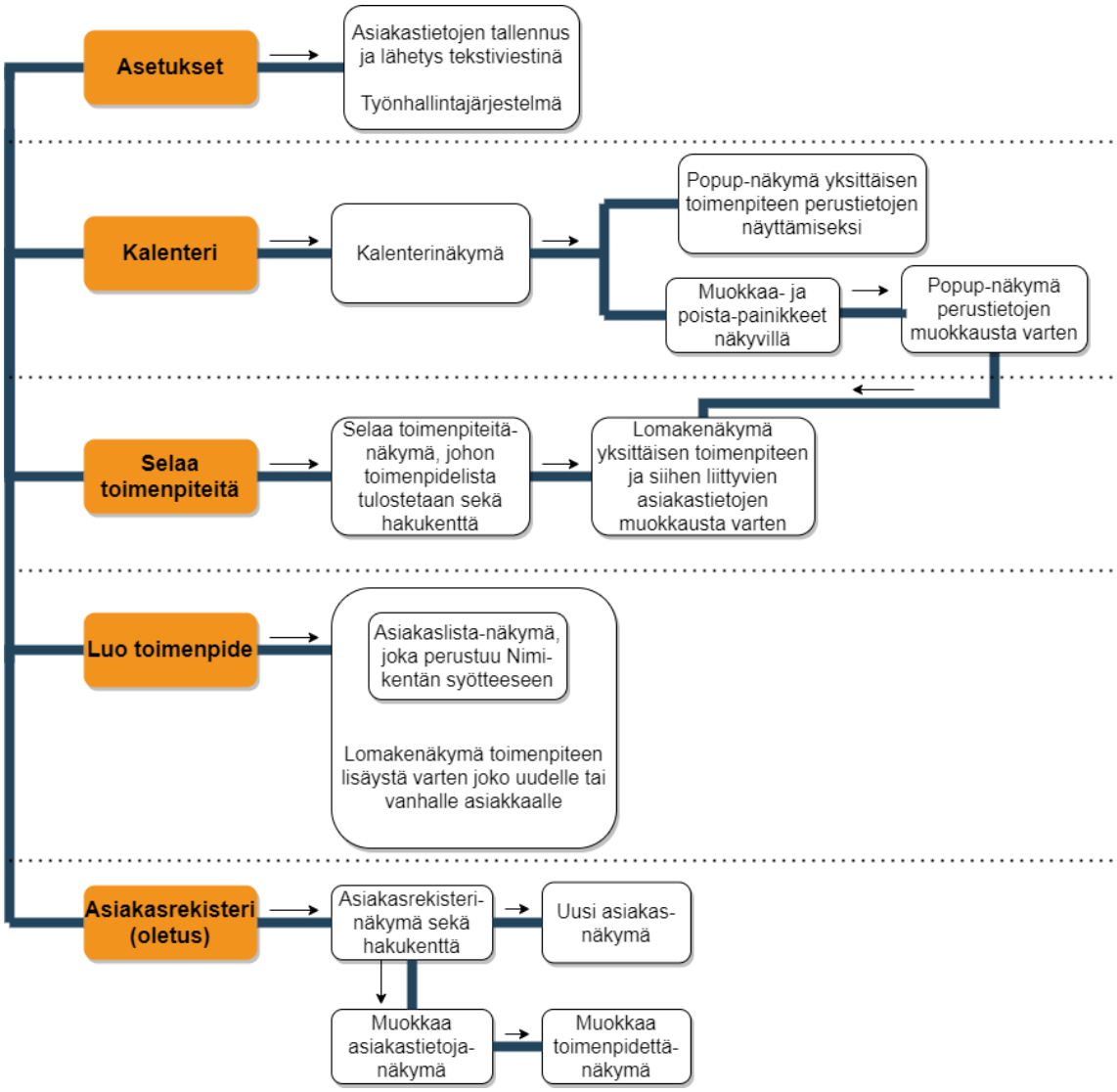
# ULKOISET LIITYNNÄT

Ulkoiset järjestelmät on esitetty kuvassa 1. Kaavion PC – ja Android-selainkäyttöliittymät edustavat käyttöliittymien ohella laitteistoliittymiä.




Kuva 1. Asiakas- ja ohjelmistovaatimusten perusteella luotu liittymäkaavio.

Käyttöliittymän valikkovalinnat ja navigointi niiden välillä on esitetty kuvassa 2. Asiakasrekisterin valinta yhden sivun sovelluksen oletusvalikkovalintana tulkittiin ohjelmistovaatimukseksi, jonka myös toimeksiantaja hyväksyi. Selainkäyttöliittymään siis avataan valmiiksi Asiakasrekisteri-valikkovalinta kirjautumisen onnistuttua.

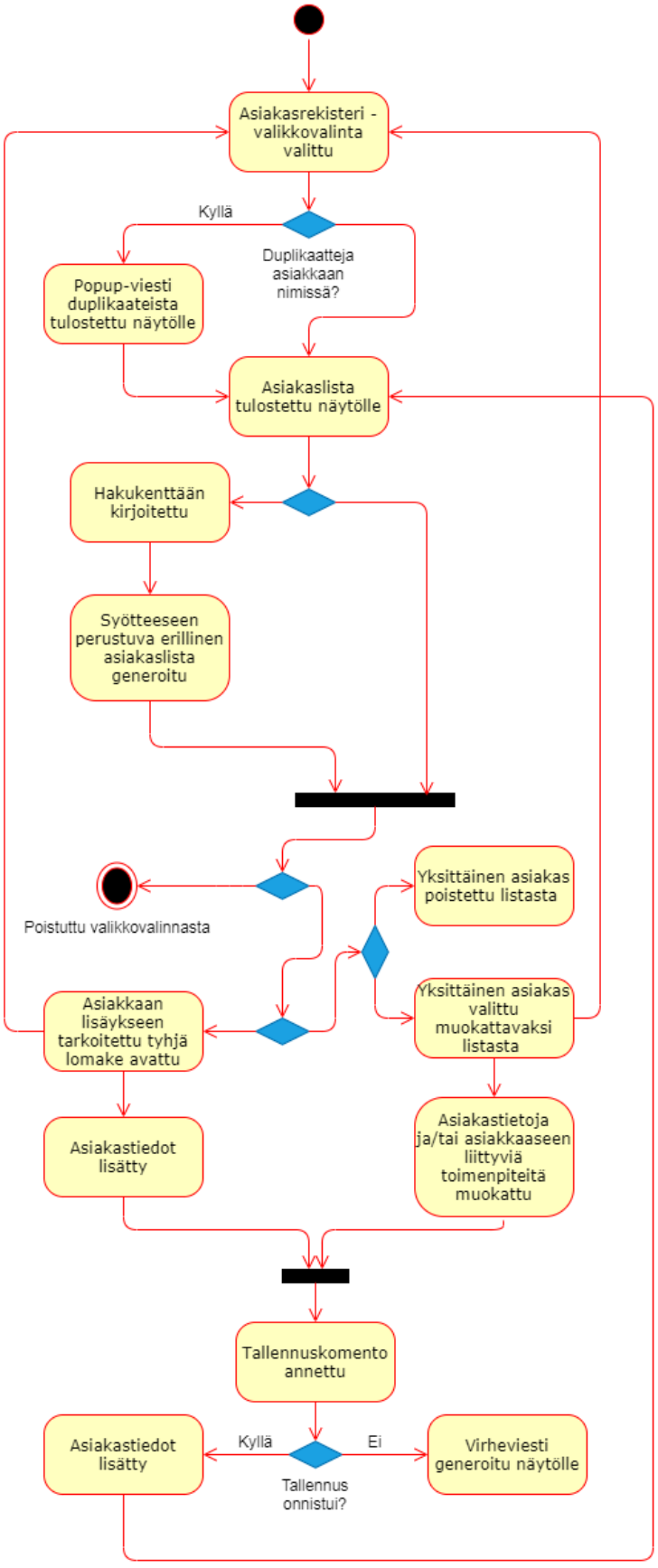



Kuva 2. Navigointi käyttöliittymässä.

## TOIMINNOT

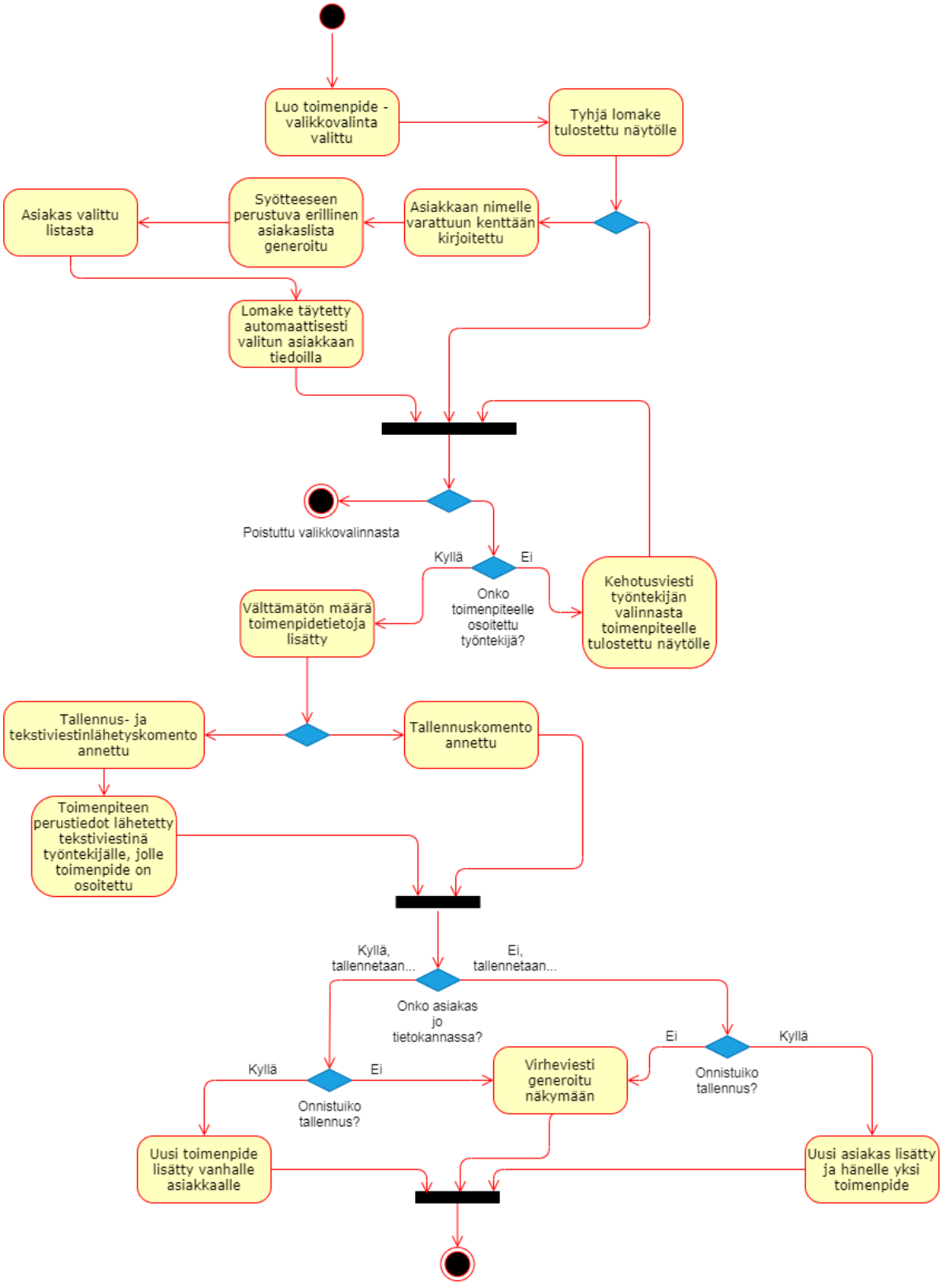
Tässä työssä tilakaavioita käytetään käyttöliittymältä – ja pohjimmiltaan koko järjestelmältä vaadittavien toimintojen kuvaamisessa. Esimerkiksi Asiakasrekisteri – valikkovalintaan liittyvät toiminnot on esitetty kuvassa 3. Paitsi kyseiseen osioon liittyvät vaatimukset (VA2-VA5 ja OV8), tilakaaviossa on esitetty myös muut vaatimusmäärittelyn aikana esiin nousseet ohjelmistovaatimukset; esimerkiksi asiakkaan lisäykseen tarkoitetulta lomakkeelta on syytä päästä takaisin asiakaslistaukseen. Tämän osion tarkoitus on kiteytettynä esittää asiakkaat listana ja joita voi etsiä hakukenttään kirjoittamalla niin, että hakusanoina käytetään joko asiakkaan nimeä, osoitetta tai asiakkaille osoitettuja toimen-

piteitä (hakusanana toimenpidetyyppi). Asiakaslistan ja hakutulosten perusteella löytyneitä asiakastietoja on päästävä muokkaamaan niin, että myös asiakkaan toimenpiteitä voidaan muokata ja/tai poistaa.



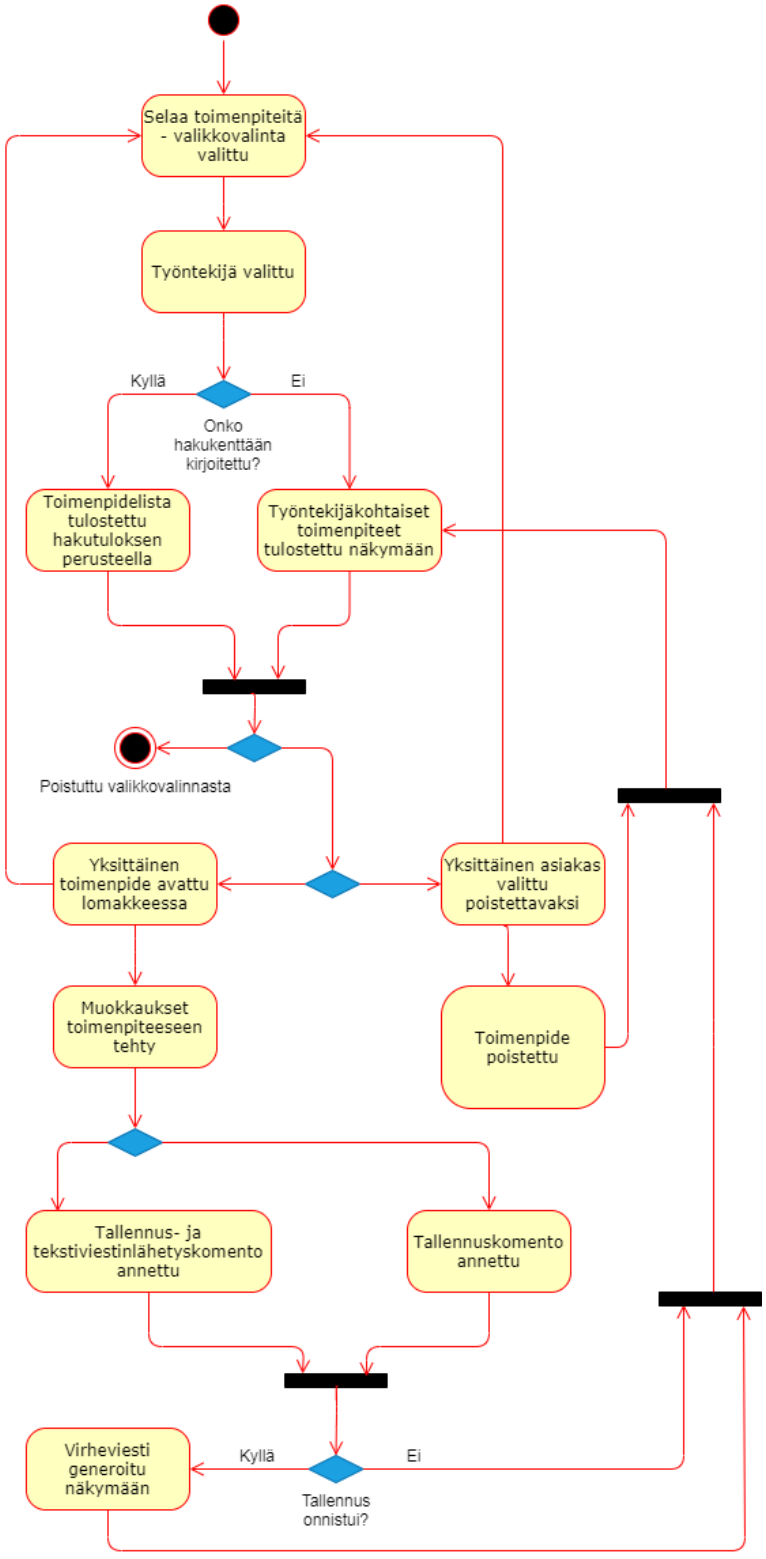

Kuva 3. Asiakasrekisteri-valikkovalinnan toiminnot asiakas- ja ohjelmistovaatimusten perusteella.

Luo toimenpide-valikkovalintaan liittyvistä vaatimuksista luotiin kuvassa 4 esitetty tilakaavio. Siihen liittyvien vaatimusten VA6-VA11 OV4-OV8 lisäksi tässäkin tilakaaviossa on kuvattu muutkin vaatimusmäärittelyn aikana esiin nousseet ohjelmistovaatimukset (tämän valikkovalinnan osalta). Tämän osion tarkoitus on lisätä uusia toimenpiteitä joko jo järjestelmään tallennetulle tai kokonaan uudelle asiakkaalle. Toimenpidetiedot on voitava tarvittaessa lähettää tekstiviestinä työntekijälle, jolle toimenpide on osoitettu.



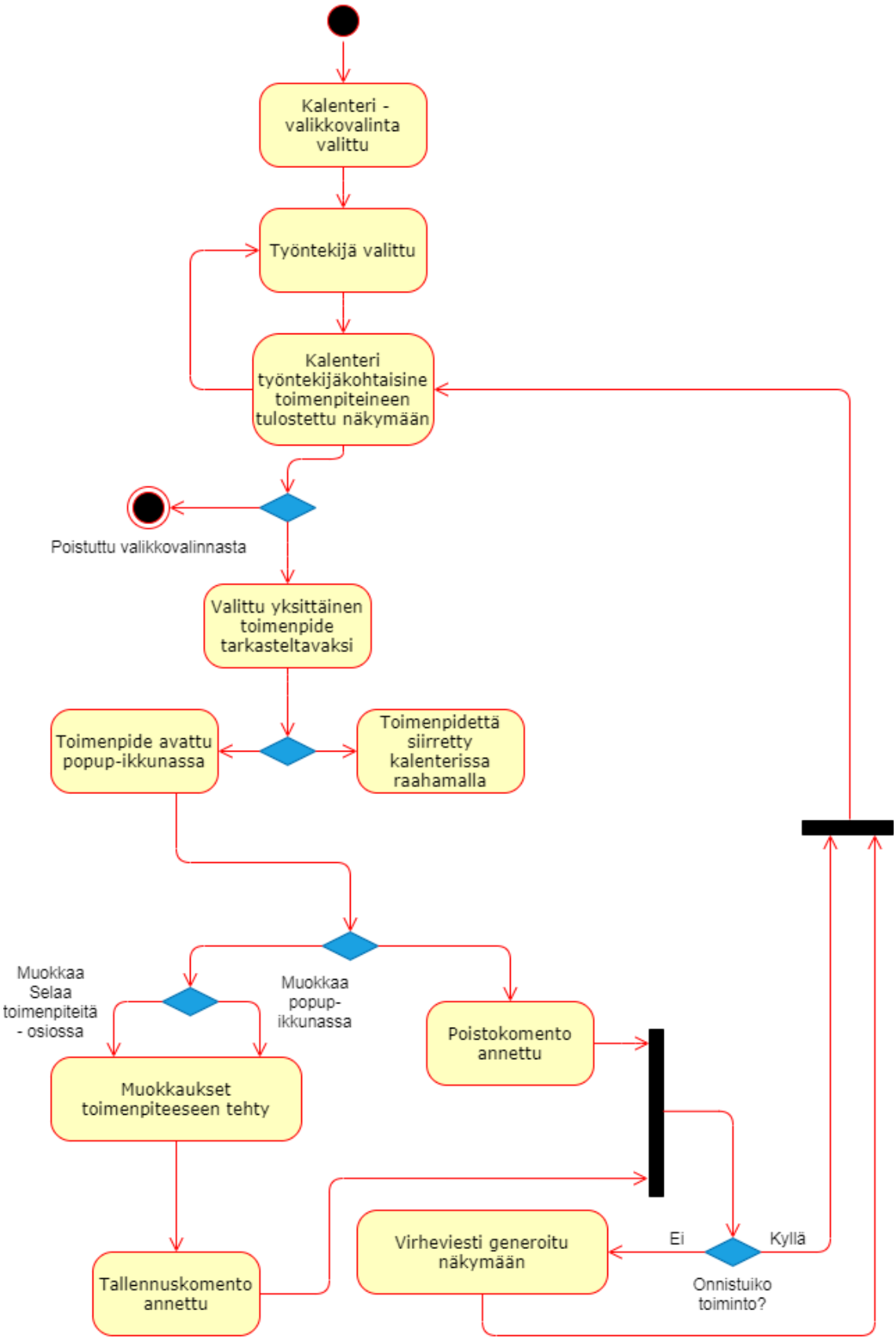

Kuva 4. Luo toimenpide-valikkovalinnan toiminnot asiakas- ja ohjelmistovaatimusten perusteella.

Selaa toimenpiteitä – valikkovalintaan liittyvien vaatimusten VA12, VA13, OV8 ja OV9 lisäksi tähän osioon löydettiin vaatimusmäärittelyssä kuvan 5 tilakaaviossa esitetyt toiminnot (ohjelmistovaatimukset). Tämän osion keskeisin tarkoitus on toimenpiteiden muokkausmahdollisuus ja se on toteutettava niin, että toimenpiteet listataan kuten Asiakasrekisteri – osiossa ja että se sisältää samanlaisen hakutoiminnallisuuden kuin mainitussa osiossa.



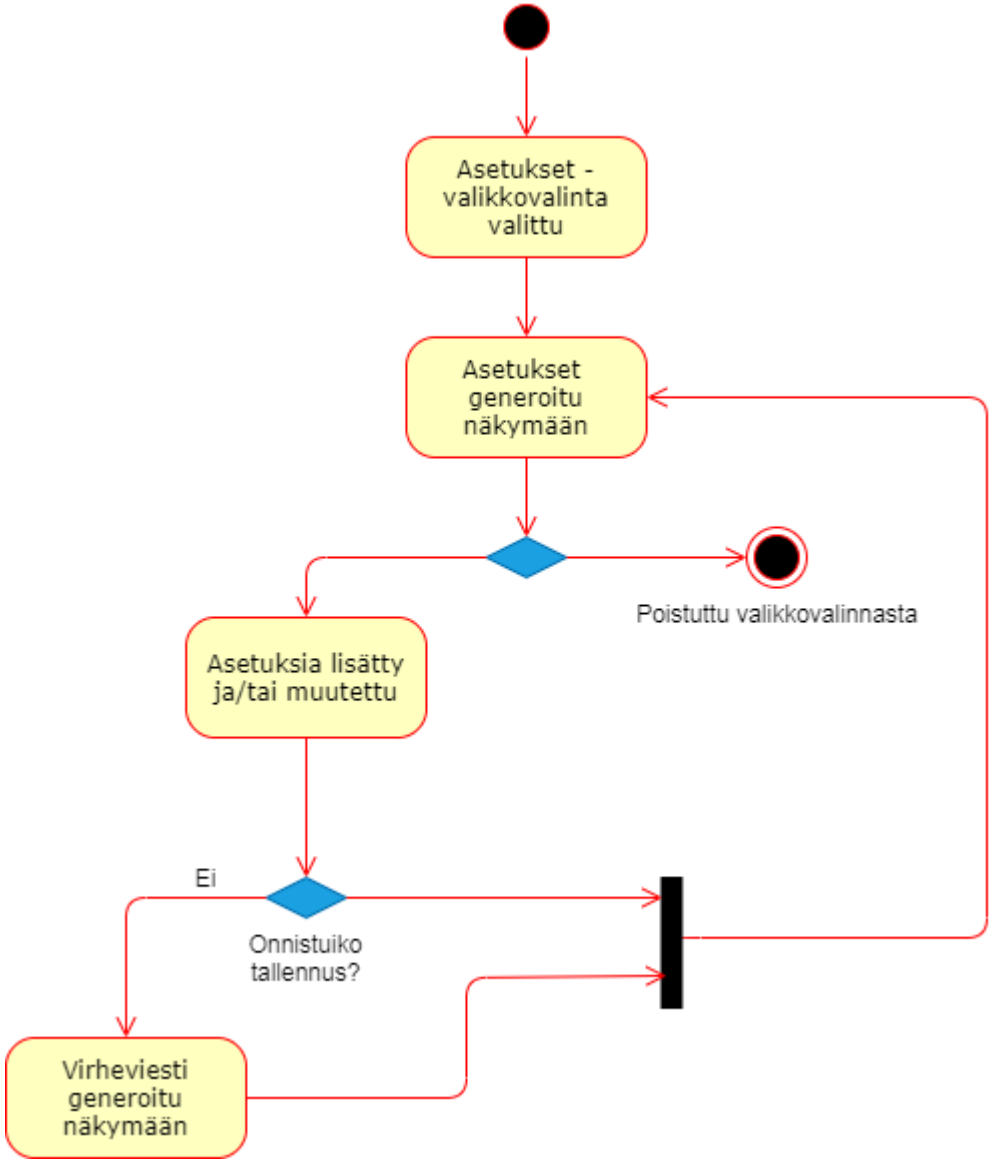

Kuva 5. Selaa toimenpiteitä – valikkovalinnan toiminnot.

Vaatimusten VA2, OV3, VA14 ja OV10 sekä muun myöhemmän vaatimusmäärittelyn perusteella luotiin kuvan 6 mukainen tilakaavio kuvaamaan Kalenteri – valikkovalinnan sisältämiä toimintoja. Tämän osion keskeisin tarkoitus on tarjota mahdollisuus aikataulujen suunnitteluun generoimalla tilatut toimenpiteet kalenteriin. Toiminnon on sisällettävä muun muassa toimenpiteiden ajankohtien ja niiden perustietojen, kuten toimenpidetyypin ja niihin liittyvien asiakkaiden yhteystietojen muokkaus. Lisäksi yksittäisen toimenpiteen muokkaukseen tarkoitetun popup-ikkunan kautta on oltava pääsy Selaa toimenpiteitä – osion muokkauslomakkeelle kyseisen toimenpiteen tiedot siihen generoituna.




Kuva 6. Kalenteri – valikkovalinnan toiminnot.

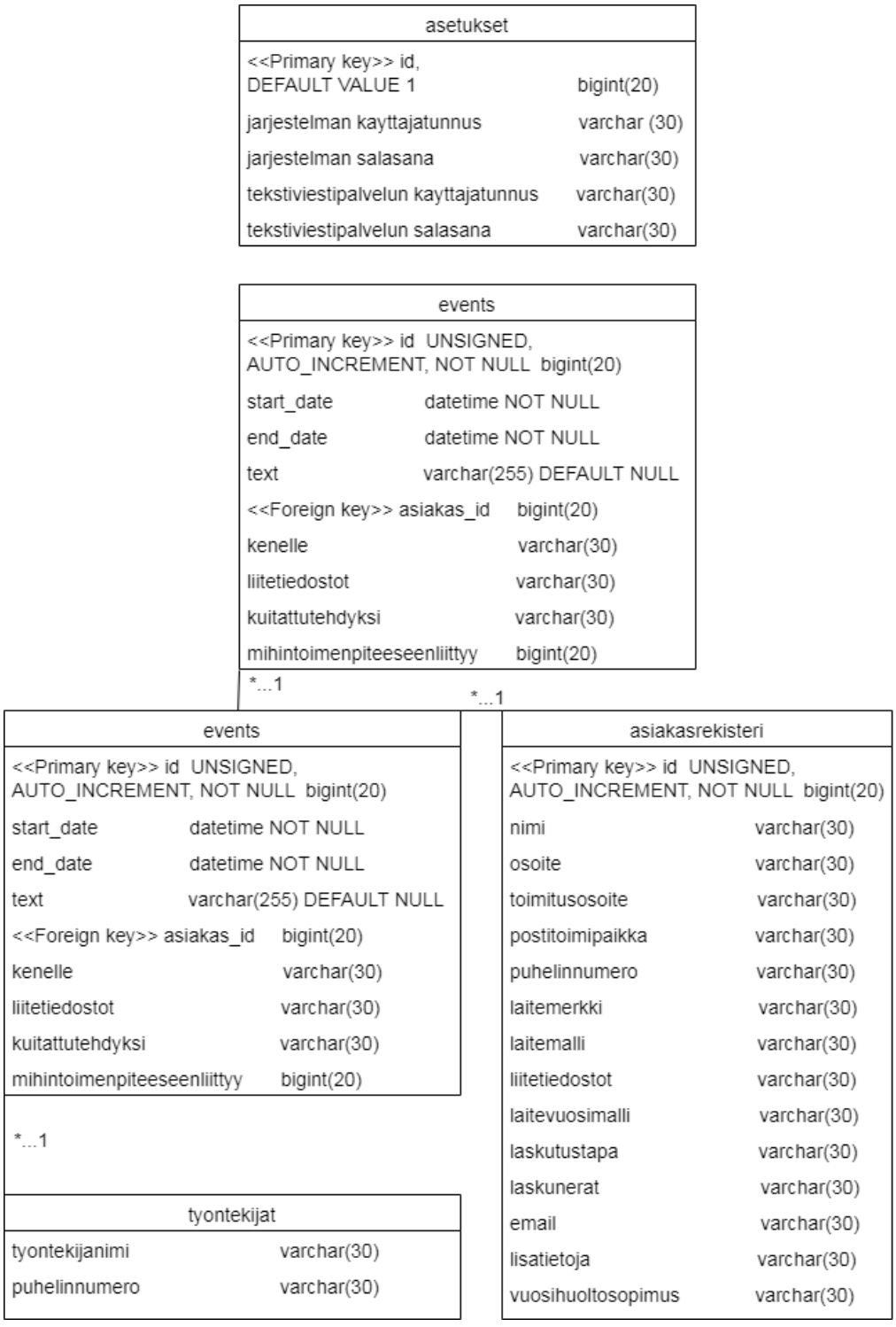
Asetukset – valikkovalintaan liittyviä toimintoja varten luotiin kuvan 7 tilakaavio. Liitteessä A tai tämän liitteen alussa listatuista vaatimuksista vain VA2 liittyy tähän osioon ja vaatimusmäärittelyn lopputuloksena muita ohjelmistovaatimuksia löytyi lähinnä vain tarve generoida jo olemassa olevat vaatimukset tämän osion näkymään ja tarve virheviestien generointiin.




Kuva 7. Asetukset – valikkovalinnan toiminnot.

# TIEDOT JA TIETOKANTA

Tietokantaan liittyen asetettiin suoraan asiakkaalta vaatimuksia niin, että esimerkiksi asiakkaan ja toimenpiteen lisäykseen tarkoitetuilla lomakkeilla tuli olla kuvan 8 asiakas -taulun mukaiset tietokentät. Kun vaatimusmäärittely täsmentyi projektin aikana niin, että Scheduler – kalenterimoduuli oli valittu kalenteritoiminnon toteuttajaksi, tietokanta määrittyi kuvan 8 *event* – taulun mukaiseen muotoon. Tähän päädyttiin lähinnä mainitun komponentin nimeämiskäytännöistä johtuen – kalenterimoduuli esimerkiksi kutsuu kaikkia toimenpiteitä events – nimisestä polusta ja käsittelee toimenpiteitä tapahtumina, joissa käytetään niiden englanninkielistä nimeä event. Koska toimenpiteiden luomisen yhteydessä tuli olla mahdollisuus paitsi työntekijän osoittamisen toimenpiteelle – myös toimenpiteen perustietojen lähettämisen kyseiselle työntekijälle, tietokantaan päätettiin määritellä myös kuvassa 8 esitetty tyontekija-niminen taulu. Autentikointiin- ja tekstiviestipalveluun liittyviin vaatimuksiin liittyen tietokantaan päätettiin määritellä myös samassa kuvassa esitetty asetukset - niminen taulu. Standardin IEEE 830 suosituksista tietokannan määrittelyyn liittyen kuvan 8 määrittely siis kuvaa tietokannan alustavat dataentiteetit ja niiden väliset suhteet, mutta muita standardin suosittelemia ominaisuuksia ei päädytty tämän laajuisessa ohjelmistossa määrittelemään.




Kuva 8. Kuvaus tietokannan dataentiteeteistä ja niiden välisistä suhteista.

# LIITE C: FUNKTIOKUTSUT JA TILATIEDOT

| Minkä tahansa valikkovalinnan päänäkymään siirryttäessä |
| --- |
| // ei olla muokkaustilassa<br>+ isModifyState = false<br>// ei olla myöskään Asiakasrekisteri-osion toimenpiteen muokkaustilassa<br>+ isModifyToimenpideOfAsiakaslomakeState = false<br>// lomakkeisiin liittyvät tiedostonimet tyhjennetään aina mihin tahansa päänäkymään<br>// siirryttäessä<br>+ earlierFileNames = []<br>// lomakkeiden kautta poistettaviksi asetetut tiedostot (tiedostonimilista) tyhjennetään<br>// vastaavasti<br>+ removables = [] |
| // tyhjennetään Luo toimenpide-osion lomakkeen kautta lisätyt tiedostot<br>+ emptyFiles()<br>// tyhjennetään Selaa toimenpiteitä-osion lomakkeen kautta lisätyt tiedostot<br>+ emptyFiles2()<br>// tyhjennetään Asiakasrekisteri-osion lomakkeen kautta lisätyt tiedostot<br>+ emptyFiles3()<br>// poistaa Selaa toimenpiteitä valikkovalinnan muokkauslomakkeen näkyvistä<br>+ hideModifyForm()<br>// tarvitaan palatessa Asiakasrekisteri-osion toimenpiteen muokkaustilasta<br>+ setMuokkaaToimenpideToAsiakasLomake()<br>// poistaa kalenteriosiossa hiiren hover-toiminnon avulla esiin tuodut toimenpiteen<br>// perustiedot<br>+ scheduler.tooltip.hide() |

Kuva 1. Funktiokutsut ja tilamuutokset mihin tahansa päänäkymään siirryttäessä.

| Kaikkiin muihin päänäkymiin paitsi Selaa toimenpiteitä siirryttäessä |
| --- |
| // asetetaan Selaa toimenpiteitä-päänäkymän pudotusvalikko oletusasetukseensa<br>+ document.getElementById("selaaToimenpiteetTyontekijanNimi").value = "Kaikki"<br>// ei olla Selaa toimenpiteitä-osion muokkaustilassa<br>+ selaaToimenpiteetModifyState = false |
| // poistetaan Selaa toimenpiteitä-osion toimenpidelista<br>+ removeToimenpiteet() |

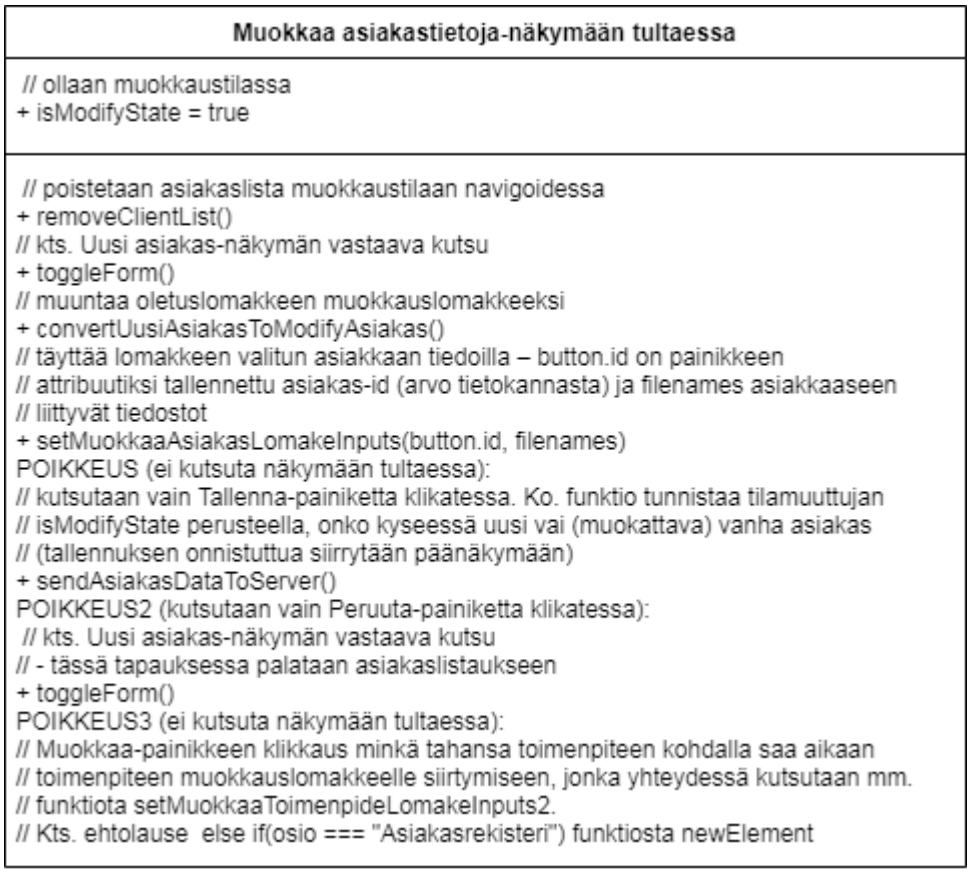
Kuva 2. Funktiokutsut ja tilamuutokset mihin tahansa päänäkymään paitsi Selaa toimenpiteitä siirryttäessä.

| Asiakasrekisteri-näkymään siirryttäessä (päänäkymä) |
| --- |
| // asetetaan hakukenttä näkyville<br>+ document.getElementById('asiakasRekisteriSearchField').style.visibility = "visible"<br>// asetetaan vakiopainikkeen teksti<br>+ document.getElementById("button1").innerHTML = "Lisää asiakas"<br>// kätkee muokkauslomakkeen<br>+ document.getElementById("myform").style.visibility = "hidden"<br>// kätkee muokkauslomakkeen tallennuspainikkeen<br>+ document.getElementById("tallennaAsiakas").style.visibility = "hidden"<br>// asettaa hakukentän valintaruudun näkyville<br>+ document.getElementById('asiakasrekisteriCheckbox').style.visibility = "visible" |
| // poistetaan vanha asiakaslista<br>+ removeClientList()<br>// generoidaan uusi asiakaslista<br>+ generateClientList()<br>//vastaa hakukentän syötteen perusteella generoitavasta listasta<br>+ setAsiakasRekisteriAutocomplete() |

Kuva 3. Funktiokutsut ja tilamuutokset Asiakasrekisteri-päänäkymään siirryttäessä.

| Uusi asiakas-näkymään siirryttäessä |
| --- |
| // ei olla muokkaustilassa<br>isModifyState = false<br>// kohdistetaan lomakkeen ylälaitaan<br>document.getElementById("button1").scrollIntoView() |
| // vastaa mm. Asiakasrekisteri-valikkovalinnan vakiopainikkeiden teksteistä ja<br>// Peruuta-painikkeen toiminnallisuudesta<br>+ toggleForm()<br>// tyhjennetään mahdolliset lomakkeeseen aiemmin jääneet tiedostot<br>+ emptyFiles3()<br>// poistetaan asiakaslista näkyvistä<br>+ removeClientList()<br>// mm. tyhjentää lomakkeen ja poistaa muokkaustilan painikkeen näkyvistä<br>+ setLuoAsiakasFormDefaults()<br>POIKKEUS (ei kutsuta näkymään tultaessa):<br>// kutsutaan vain Tallenna-painiketta klikatessa<br>// (tallennuksen onnistuttua siirrytään päänäkymään)<br>+ sendAsiakasDataToServer() |

Kuva 4. Funktiokutsut ja tilamuutokset Asiakasrekisteri-osion Uusi asiakas-näkymään siirryttäessä.




Kuva 5. Funktiokutsut ja tilamuutokset Asiakasrekisteri-osion Muokkaa asiakas-näkymään siirryttäessä.

**Asiakasrekisteri-osion Muokkaa toimenpidettä-näkymään tultaessa**

```
// ollaan muokkaustilassa
isModifyState = true
// ollaan lisäksi Asiakasrekisteri-osion toimenpiteen muokkaustilassa
isModifyToimenpideOfAsiakaslomakeState = true
```

```
Kts. ehtolause else if(osio === "Asiakasrekisteri") funktiosta newElement

// poistaa muokkaa asiakastietoja-lomakkeelle generoidut toimenpiteet
+ removeToimenpiteet()
// mm. tyhjentää muokkaa asiakastietoja-lomakkeen tietokentät
+ setLuoAsiakasFormDefaults()
// muokkaa lomaketta edelleen toimenpiteen muokkaukseen sopivaksi
+ muokkaaAsiakasLomakeToMuokkaaToimenpide()
// täyttää lomakkeen toimenpiteeseen ja siihen liittyvillä
// asiakastiedoilla
+ setMuokkaaToimenpideLomakeInputs2(button.id, button.value, filenames)
POIKKEUKSET(ei kutsuta näkymään tultaessa):
// tallennetaan toimenpidelomakkeen tiedot asiakastietoineen ja lähetetään sen
// perustiedot työntekijälle, jolle toimenpide on osoitettu
// (tallennuksen onnistuttua siirrytään päänäkymään)
+ sendClientDataSMS("Asiakasrekisteri")
// tallennetaan toimenpidelomakkeen tiedot asiakastietoineen
// kutsutaan vain Tallenna-painiketta klikatessa. Ko. funktio tunnistaa tilamuuttujan
// isModifyState perusteella, onko kyseessä uusi vai (muokattava) vanha asiakas
// (tallennuksen onnistuttua siirrytään päänäkymään)+ sendModifiedToimenPideDataToS
POIKKEUKSET2(ei kutsuta näkymään tultaessa):
// kts. Uusi asiakas-näkymän vastaava kutsu
// - tässä tapauksessa palataan asiakaslomakkeelle
+ toggleForm()
```

Kuva 6. Funktiokutsut ja tilamuutokset Asiakasrekisteri-osion Muokkaa toimenpidettä-näkymään siirryttäessä.

**Luo toimenpide-näkymään siirryttäessä**

```
// asetetaan näkymän lomake näkyville
+ showForm()
 // tyhjennetään lomakkeen tietokentät
+ setLuoToimenpideFormDefaults()
// asetetaan nimi-kentän syötteen perusteella generoitavaan asiakaslistaan liittyvä
// toiminto valmiiksi
+ autocomplete(document.getElementById("luoToimenpideTilaajanNimi"),customers)
// kohdistetaan lomakkeen yläosaan
+ document.getElementById("luotoimenpidE").scrollIntoView()
// tämän funktiokutsun rajoittamista vain Asetukset-osioon on syytä tutkia
+  getAllSettings()
POIKKEUKSET (ei kutsuta näkymään tultaessa):
// tallentaa toimenpiteen, asiakastiedot (uusi asiakas) tai mahdolliset muutokset
// asiakastietoihin (vanha asiakas) sekä lähettää perustiedot SMS:nä pudotusvalikosta
// valitulle työntekijälle. Tallennuksen jälkeen palataan oletusnäkymään
// (Asiakasrekisterin päänäkymä)
sendClientDataSMS("Luo toimenpide")
// muutoin sama toiminto kuin edellisessä, mutta ilman SMS-toimintoa
+ sendLuoToimenPideDataToServer()
```

Kuva 7. Funktiokutsut ja tilamuutokset Luo toimenpide-osion lomakenäkymään siirryttäessä.

**Selaa toimenpiteitä-päänäkymään siirryttäessä**

```
// kaksi ensimmäistä tilan asetusta varmistaa, että pudotusvalikko asetetaan näkyviin
+ document.getElementById("selaaToimenPiteetButtons").style.visibility = "visible"
+ document.getElementById("selaaToimenPiteetButtons").style.display = "block"
// asetetaan hakukenttä näkyville
var searchField = document.getElementById('selaaToimenpiteitaSearchField');
+ searchField.style.visibility = "visible";
```

```
// tämä funktio vastaa toimenpiteiden näyttämisestä sen mukaan mikä on pudotusvalikon
// sekä hakukenttään liittyvän valintaruudun arvo
showToimenpiteet()
// showToimenpiteet-funktion kautta kutsutaan myös funktiota
// sendEmployerNameToServer, jonka tehtävä on lähettää pudotusvalikon työntekijänimi
// palvelimelle, joka palauttaa nimeä vastaavan toimenpidelistan selainkoodin globaaliin
// muuttujaan (ko. lista palautetaan erikseen kalenterimoduulille)
// tätä kutsua tarvitaan muokkaustilasta päänäkymään palatessa
// (tyhjentää lomakkeesta tiedostot)
removeAllFilesForm();
// tyhjentää Selaa toimenpiteitä-osion lomakkeen tietokentät
setMuokkaaToimenpideFormDefaults()
// tällä ehtolauseella varmistetaan, että muokkaustilasta päänäkymään siirryttäessä
// ei generoida työntekijäkohtaisia toimenpiteitä, vaan kaikki toimenpiteet
// (samaa pudotusvalikkoa käytetään myös muokkauslomakkeessa)
if(!selaaToimenpiteetModifyState){
   $("#selaaToimenpiteetTyontekijanNimi").val("Kaikki");
   }
// tämän funktiokutsun rajoittamista vain Asetukset-osioon on syytä tutkia
+  getAllSettings()
POIKKEUS (ei kutsuta näkymään siirryttäessä):
// Muokkaa-painikkeen klikkaus minkä tahansa toimenpiteen kohdalla saa aikaan
// muokkauslomakkeelle siirtymiseen, jonka yhteydessä kutsutaan mm. funktiota
// setMuokkaaToimenpideLomakeInputs. Kts. ehtolause  if(osio === "Selaa toimenpiteet")
// funktiosta newElement
```

Kuva 8. Funktiokutsut ja tilamuutokset Selaa toimenpiteitä-osion listausnäkymään siirryttäessä.

**Selaa toimenpiteitä-muokkausnäkymään siirryttäessä**

```
// poistetaan päänäkymän pudotusvalikko näkyvistä
var selaaToimenPiteetButtons = document.getElementById("selaaToimenPiteetButtons")
selaaToimenPiteetButtons.style.display = "none"
// poistetaan hakukentän valintaruutu näkyvistä
var checkBox = document.getElementById("selaaToimenpiteetCheckbox")
checkBox.style.visibility = "hidden"
```

```
Kts. ehtolause if(osio === "Selaa toimenpiteet") funktiosta newElement

// poistetaan Selaa toimenpiteitä-päänäkymän toimenpiteet
+removeToimenpiteet3()
// poistetaan hakukenttä näkyvistä ja asetetaan muokkauslomake näkyville
toggleForm3()
// täytetään muokkauslomake valitun toimenpiteen tiedoilla
+setMuokkaaToimenpideLomakeInputs(button.id, button.value, filenames)
```

Kuva 9. Funktiokutsut ja tilamuutokset Selaa toimenpiteitä-osion Muokkaa toimenpidettä-näkymään siirryttäessä.

**Kalenterinäkymään siirryttäessä**

```
// tämän funktion kautta kutsuttava sendEmployerNameToServer-funktio vastaa
// Selaa toimenpiteitä-osiossa käytetyn työntekijäkohtaisen suodatuksen
// synkronoimisesta Kalenteriosioon.
+calendarExtra()


POIKKEUS (kutsutaan vain Selaa toimenpide-osion muokkauslomakkeelle siirryttäessä):
//funktiokutsussa scheduler.attachEvent("onLightboxButton", function(button_id, node, e)
// määritellään Avaa lomakkeessa-tekstillä varustetun painikkeen toiminto; avataan
// tarkasteltava toimenpide Selaa toimenpiteitä-osion muokkauslomakkeessa
```

Kuva 10. Funktiokutsut ja tilamuutokset Kalenteri-osion päänäkymään siirryttäessä.

**Kalenterinäkymän funktio initAndLoadScheduler**

```
// funktiota kutsutaan aina sivun latauksen yhteydessä sekä aina kun dataan tehdään
// muutos, lisäys tai poisto muissa osioissa .Tämän tarpeellisuus kannattaa tutkia, sillä
// kalenteriosion päänäkymään siirryttäessä kaikki data päivitetään joka tapauksessa

// esim. scheduler.locale.labels.section_toimitusosoite   = "Toimitusosoite"
// asettaa muokkausikkunan tekstikentälle halutun nimen

//  Lohkossa scheduler.config.lightbox.sections määritellään, minkä nimisistä
events-objektilistan (globaali muuttuja) sisältämien objektien attribuuttien arvoista
generoidaan mihinkin muokkausikkunan tekstikenttään

// seuraavan kaltaisissa lohkoissa poistetaan muokkausikkunan tekstikentästä
// kirjoitusoikeus (tässä tapauksessa nimi-kenttä)
scheduler.attachEvent("onLightbox", function(){
        var section = scheduler.formSection("nimi");
        section.control.disabled = true;
 });

// allaolevan kutsuketjun merkitys on siinä, että esim. kalenterin kautta tehtävä muutos
// tapahtuman suoritusajankohtaan päivittyisi mahdolliseen päivän viimeiseen
// tapahtumaan eikä uutta tapahtumaa generoitaisi sen kanssa limittäin (tai jopa alle).
// kutsu getClients näet päivittää uusien toimenpiteiden alustavaan ajoitukseen tarvittavat
// muuttujat
scheduler.attachEvent("onEventChanged", function(id,ev){
        getClients();
)}

// kalenterimoduulin laajempi ohjeistus Node.js-ympäristöön liittyen sivulla (viitattu 11.3.2021)
// https://docs.dhtmlx.com/scheduler/howtostart_nodejs.html
```

Kuva 11. Kalenterin lataava ja päivittävä funktio initAndLoadScheduler.

# LIITE D: TESTAUS KÄYTTÖÖNOTTOA VARTEN

## LUO TOIMENPIDE-OSIO

**Uuden toimenpiteen lisäys vanhalle asiakkaalle**

1. Tarkista, että autocomplete-listan kautta valittavan asiakkaan tiedot generoidaan oikein lomakkeelle sekä tapauksessa, jossa asiakas valitaan listasta kursorin avulla, että tapauksessa, jossa asiakas valitaan nuolinäppäimiä käyttämällä.

Kaikki testattu tähän saakka ja tarvittavat korjaukset tehty 12.8.2020.

**Uuden toimenpiteen lisäys uudelle asiakkaalle**

1. Avaa Luo toimenpide-osion kautta avautuva lomake

2. Tarkista, että lomakkeelta on pääsy takaisin Asiakasrekisteri-osion asiakaslistaan Asiakasrekisteri-valikkovalintaa klikkaamalla ja että asiakaslista generoituu asianmukaisesti.

3. Täytä tekstikentät kuhunkin aiheeseen sopivalla uniikilla tekstillä niin, että toimenpide osoitetaan asiakkaalle, jota ei löydy autocomplete-listasta, valitse toimenpiteen suorittaja pudotusvalikosta, lisää toimenpiteelle yksi tiedosto ja klikkaa Tallenna toimenpide - painiketta.

4. Tarkista, että tietokantaan tallentuu uusi toimenpide events-tauluun niin, että taulun text-sarakkeessa näkyy lomakkeella valittu toimenpidetyyppi. Lisäksi tiedostonimi omaan sarakkeeseensa ja toimenpiteen suorittaja kenelle-sarakkeeseen. Tässä vaiheessa sarake ”kuitattu tehdyksi” pitäisi olla muotoa ”0000-00-00”. Sarake ”mihintoimenpiteeseenliittyy” on oltava tyhjä, sillä se liittyy vain vuosihuoltosopimus-tyyppisiin toimenpiteisiin, jotka on generoitu vuoden päähän edellisestä. Tarkista myös, että asiakasrekisteri-taulussa on uusi tietue vastaavilla sarakearvoilla kuin lomakkeelle lisättiin (asiakastietueella ei saa näkyä tässä vaiheessa liitetiedostoja). Lisäksi on tarkistettava, että mainitun tietueen id-arvo näkyy lisätyn toimenpiteen asiakasrekisteri_id-sarakkeessa.

4.1 Varmista, että toimenpiteen suorittaja tallentuu molemmilla vaihtoehdoilla (Joonas/Vellu) tietokantaan oikein.

5. Jos samalle päivälle ei ole lisätty toimenpiteitä, niin lisätyllä toimenpiteellä pitäisi olla aloitusaika 07:00 ja lopetusaika 08:00. Jos taas samalle päivälle on jo lisätty toimenpiteitä eikä päivän viimeisen toimenpiteen lopetusaika ole yli 20:00, niin lisätyn toimenpiteen pitäisi saada aloitusaika, joka on päivän viimeisen toimenpiteen lopetusaika ja lopetusajaksi pitäisi tulla tunti aloitusajasta. Jos taas viimeisen toimenpiteen lopetusaika on yli 20:00, lisätylle toimenpiteelle tarkistetaan edellä mainitut ehdot päiviä niin kauan lisäten, kunnes löytyy päivä, josta mainitulla ehdoilla löytyy tilaa uudelle toimenpiteelle. Tarkista toimivuus sekä ”tyhjillä” päivillä, että päivillä, joissa on toimenpide ajoitettuna lopetusajoilla 20:00 ja 20:05.

6. Lisää uusi toimenpide (aina uudelle asiakkaalle) niin monta kertaa, kunnes lomakkeen kaikki radio button - toiminnot on käyty läpi. Tarkista kullakin kerralla, että tietokannan events-osioon tallentuu toimenpidetyyppi ja uudelle asiakkaalle asiakasrekisteri-tauluun kutakin radio button – valintaa vastaava arvo.

7. Lisää uusi toimenpide (aina uudelle asiakkaalle) niin monta kertaa, että kaikki seuraavat tapaukset ovat testatut: Uuteen toimenpiteeseen liittyy 1, 2 ja 3 (uniikkia) tiedostoa. Tarkista kullakin kerralla, että events-taulun liitetiedostot-sarakkeeseen tallentuu toimenpiteen tiedostonimet (1-3 kpl) ja itse tiedostot palvelimen files-kansioon.

8. Tarkista, että Liitetiedostot-osion Tyhjennä-painike poistaa näkymästä osion kautta tallennettaviksi valitut tiedostot tapauksissa 1, 2 ja 3 tiedostoa.

9. Tarkista, että Tyhjennä-painikkeen klikkausten jälkeen lisätyt tiedostot tallentuvat sekä tiedostonimiksi tietokantaan, että itse tiedostoiksi palvelinkoodin files-kansioon.

10. Tarkista, että sekä uudelle, että vanhalle asiakkaalle lisätyn toimenpiteen tiedoista asiakkaan nimi, osoite, toimitusosoite, puhelinnumero, laitteen merkki ja lisätietoja-kentän tiedot lähetetään asetusten kautta määriteltyyn puhelinnumeroon tekstiviestinä.

Kaikki testattu tähän saakka ja tarvittavat korjaukset tehty 12.8.2020.

## ASIAKASREKISTERI-OSIO

**Uuden asiakkaan lisäys, tietojen generointi muokkauslomakkeelle sekä asiakastietojen poisto**

1. Avaa Lisää asiakas-painikkeen kautta avautuva lomake.

2. Tarkista, että Lisää asiakas-painikkeen kautta avautuvalta lomakkeelta on pääsy takaisin Asiakasrekisteri-osion asiakaslistaan Peruuta-painiketta klikkaamalla.

3. Täytä tekstikentät kuhunkin aiheeseen sopivalla uniikilla tekstillä ja klikkaa Tallenna-painiketta.

4. Tarkista, että tietokantaan tallentuu uusi asiakas asiakasrekisteri-tauluun tekstikenttiä vastaavilla arvoilla.

4.1 Tarkista, että lisätty asiakas näkyy uutena rivinä Asiakasrekisteri-osion asiakaslistassa Muokkaa- ja Poista-painikkeineen niin, että asiakkaan nimi, toimitusosoite ja paikkakunta tulostuvat riville.

4.2 Tarkista Muokkaa-painiketta klikkaamalla, että kyseisen asiakkaan tiedot generoidaan lomakkeelle vastaavasti kuin tietokannassa. Kun tiedostoja ei ole lisätty, Liitetiedostot-osion yläpuolella tulee olla otsikkoteksti ”Ei Asiakasrekisteri-osion kautta lisättyjä tiedostoja”.

5. Lisää uusi asiakas niin monta kertaa, kunnes lomakkeen kaikki radio button - toiminnot on käyty läpi. Tarkista kullakin kerralla, että tietokannan asiakasrekisteri-osioon tallentuu kutakin radio button – valintaa vastaava arvo.

5.1 Tarkista kussakin tapauksessa, että radio button-valinnat generoidaan oikein Muokkaa-painikkeen kautta avautuvalle lomakkeelle.

6. Lisää uusi asiakas niin monta kertaa, että kaikki seuraavat tapaukset ovat testatut: Uuteen asiakkaaseen liittyy 1, 2 ja 3 tiedostoa. Tarkista kullakin kerralla, että asiakasrekisteri-taulun liitetiedostot-sarakkeeseen tallentuu asiakkaan tiedostonimet (1-3 kpl) ja itse tiedostot palvelimen files-kansioon.

6.1 Tarkista kussakin tapauksessa, että tiedostot generoidaan hyperlinkkeineen Muokkaa-painikkeen kautta avautuvalle lomakkeelle. Kukin hyperlinkki tulee muodostua tiedoston nimestä. Samalla rivillä tulee olla myös Poista-painike. Tiedostojen yläpuolella tulee näkyä otsikkoteksti ”Asiakasrekisteri-osion kautta lisätyt tiedostot”. Tarkista myös, että kukin hyperlinkki toimii. Tarkista lisäksi, että useita asiakkaita tiedostoineen lisättäessä kullekin asiakkaalle generoituu juuri oikeat tiedostot Muokkaa asiakas-lomakkeelle.

6.2 Tarkista, että tiedoston poistotoiminto toimii niin, että tiedostorivi deletoituu paitsi näkymästä, myös tietokannasta sekä itse tiedosto files-kansiosta.

6.3 Tarkista, että Liitetiedostot-osion Tyhjennä-painike poistaa näkymästä osion kautta tallennettaviksi valitut tiedostot tapauksissa 1, 2 ja 3 tiedostoa.

6.4 Tarkista, että Tyhjennä-painikkeen klikkausten jälkeen lisätyt tiedostot tallentuvat sekä tiedostonimiksi tietokantaan, että itse tiedostoiksi palvelinkoodin files-kansioon.

6.5 Tilanteessa, jossa uudelle asiakkaalle ei ole osoitettu toimenpiteitä, varmista, että asiakaslistan yksittäisen asiakkaan poistotoiminto toimii niin, että asiakas deletoituu paitsi Asiakasrekisteri-osion asiakaslistasta, myös asiakasrekisteri-tietokantataulusta. Samassa yhteydessä on poistuttava myös kaikki asiakkaalle osoitetut tiedostot palvelinkoodin files-kansiosta.

6.6 Varmista, että asiakaslomakkeesta toiseen siirtyminen todella generoi tiedot oikein eikä edellisen asiakkaan tiedot vaikuta tietojen generointiin.

Kaikki testattu ja tarvittavat korjaukset tehty 10.8.2020.

**Asiakastietojen muokkaus**

1. Poimi yksittäinen asiakas listasta Muokkaa-painiketta klikkaamalla ja muuta sekä jokaista tekstikenttää, että kaikkia radio button-valintoja niin, että kaikki lomakkeen tiedot toimenpiteitä lukuunottamatta muuttuvat. Sen jälkeen tallenna asiakastiedot.

1.1 Tarkista, että muutetut asiakastiedot päivittyvät vastaavaan asiakasrekisteri-taulun sarakkeisiin.

1.2 Avaa kyseiset asiakastiedot uudestaan ja varmista, että muokatut asiakastiedot generoituvat Muokkaa-painikkeen kautta avautuvalle lomakkeelle.

2. Tee vielä muutoksia kaikkiin radio button-valintoihin niin, että käyt läpi kaikki mahdolliset radio button-arvot ja tarkista niidenkin vastaavuus asiakasrekisteri-taulun tietueessa.

2.1 Varmista jokaisen muokkauksen jälkeen, että muokkaukset generoituvat oikein Muokkaa-painikkeen kautta avautuvalle lomakkeelle.

3. Toista kohdan ”Uuden asiakkaan lisäys, tietojen generointi muokkauslomakkeelle sekä asiakastietojen poisto” kohdat 6. – 6.4 ja 6.6 niin, että ei lisätä uusia asiakkaita, vaan tehdään muutoksia Muokkaa-painikkeen kautta avautuvan lomakkeen avulla samalle asiakkaalle.

Kaikki testattu ja tarvittavat korjaukset tehty 12.8.2020.

**Asiakkaalle osoitetun toimenpiteen muokkaus**

1. Lisää yksittäiselle asiakkaalle Huoltokäynti-toimenpide ja avaa kyseisen asiakkaan tiedot asiakaslistasta Muokkaa-painikkeen avulla.

1.1 Tarkista, että lomakkeen Asiakkaaseen liittyvät toimenpiteet-otsikon alla.

näkyy juuri lisätty toimenpide punaisella taustavärillä ja että sen tiedoissa mainitaan ensimmäisenä päivämäärä ja kellonaika (toimenpiteen aloitusaika), seuraavina toimenpidetyyppi, asiakkaan nimi ja toimitusosoite.

1.2 Tarkista, että toimenpiderivin vasemmassa päässä on sinipohjainen Muokkaa-tekstillä varustettu painike ja oikeanpuoleisessa päässä punapohjainen Poista-tekstillä varustettu lomake.

1.3 Varmista, että toimenpiteen muokkaukseen tarkoitetun lomakkeen kautta on pääsy asiakastietojen muokkaukseen tarkoitetulle lomakkeelle Peruuta-tekstin omaavan painikkeen kautta sekä saman painikkeen kautta edelleen asiakastietojen muokkaukseen tarkoitetulta lomakkeelta asiakaslistaukseen (Asiakasrekisteri-osion oletusnäkymä).

2. Klikkaa toimenpiderivin Muokkaa-painiketta ja tarkista, että toimenpiteen tiedot generoituvat vastaavasti kuin asiakasrekisteri- ja events-tietokantatauluissa.

3. Toista lomakkeelle soveltuvin osin Asiakastietojen muokkaus-otsikon alta löytyvät testit. Testeissä on varmistettava, että saman ja eri asiakkaiden toimenpiteiden välillä (lomake) liikkumisen yhteydessä edellisen toimenpiteen tiedot eivät vaikuta seuraavan toimenpiteen tietojen generointiin.

3.1 Varmista myös, että pudotusvalikko, jolla määrätään toimenpiteen suorittaja toimii niin, että ensinnäkin sen muutokset päivittyvät tietokantaan ja että se generoidaan oikein muutosten jälkeen lomakkeelle.

4. Lisää sen jälkeen samalle asiakkaalle vuosihuoltokäynti ja varmista, että se näkyy asiakaslomakkeella samalla oranssilla, joka on Luo toimenpide-lomakkeen toimenpidetyypin määrittävissä radio buttoneissa.

4.1 Lisää sen jälkeen samalle asiakkaalle vuosihuoltosopimuskäynti / vaihda olemassa olevan toimenpiteen tyyppi vuosihuoltokäynniksi ja varmista, että se näkyy asiakaslomakkeella keltapohjaisena.

4.2 Lisää sen jälkeen samalle asiakkaalle lähetä tuote-tyyppinen toimenpide / vaihda olemassa olevan toimenpiteen tyyppi mainitunlaiseksi ja varmista, että se näkyy asiakaslomakkeella sinipohjaisena.

4.3 Lisää sen jälkeen samalle asiakkaalle soittopyyntö-tyyppinen toimenpide / vaihda olemassa olevan toimenpiteen tyyppi mainitunlaiseksi ja varmista, että se näkyy asiakaslomakkeella sinipohjaisena.

4.4 Tarkista kokeilemalla, että Asiakastietojen muokkaukseen tarkoitetun

lomakkeen kautta avautuvan toimenpiteen muokkaukseen tarkoitetun lomakkeen kautta tehtävät muutokset (asiakkaan nimi, osoitteet, paikkakunta, puhelinnumero, laitteen merkki sekä lisätietoja-kentän tiedot) päätyvät toimenpiteen työntekijälle osoitettuun matkapuhelinnumeroon.

4.5 Tarkista, että toimenpiteen Kuitattu tehdyksi-kohdan muutokset.

päivittyvät tietokantaan ja että muutokset generoidaan lomakkeelle, kun se avataan uudestaan. Valmiiksi kuitatun toimenpiteen tulee näkyä tyypistä riippumatta asiakastietolomakkeella vihreäpohjaisena ja sen yhteydessä tulee näkyä myös kuittauspäivämäärä – sekin vihreäpohjaisena. Lisäksi jos kyseessä on vuosihuoltosopimuskäynti, niin kuitattaessa se tehdyksi, täytyy järjestelmän generoida uusi samanlainen toimenpide vuoden päähän edellisestä noin kuukauden tarkkuudella. Järjestelmän tulee sijoittaa uusi toimenpide enintään vuoden ja kuukauden päähän siitä, kun edellinen toimenpide kuitataan valmiiksi. Varmista myös, että vuoden päähän generoitu vuosihuoltosopimuskäynti täyttää em. ehdot kuitattaessa se tehdyksi.

Kaikki testattu ja tarvittavat korjaukset tehty 18.8.2020.

**Hakukentän testaus**

1. Varmista, että hakukenttä ehdottaa listaa perustuen joko asiakkaan nimeen, paikkakuntaan tai toimenpiteeseen. Lista on ilmestyttävä jo ensimmäisen kirjaimen kirjoittamisesta lähtien ja sen on koostuttava asiakkaan nimestä, paikkakunnasta ja toimenpiteestä (toimenpiteistä) koostuvista olioista, joiden mikä tahansa attribuutti (asiakkaan nimi, paikkakunta tai toimenpide) voi toimia syötetyn merkkijonon vertailukohteena. Syötetty merkkijono pitää näkyä listaolion sen attribuutin kohdalla lihavoituna, johon se täsmää.

Hakutoiminto siis listaa esimerkiksi syötteellä ”Hu” kaikki asiakkaat, joille on kirjattu huoltotoimenpiteitä – tai jos asiakkaalla on Hu-alkuinen etu- tai sukunimi tai paikkakunta, niin listaan lisätään siinä tapauksessa myös asiakkaat, joille ei ole osoitettu huoltotoimenpiteitä. Haun täytyy toimia niin, että jos syötettä vastaava merkkijono löytyy sukunimestä (ei vain etunimestä), niin toiminto huomio lisää myös sellaiset asiakkaat ehdotettavaan listaan.

2. Varmista, että hakutoiminnon ehdottamasta listasta sivulla näkyy valinnan jälkeen ainoastaan valittu asiakas ja että Muokkaa-painikkeen klikkaus johtaa kyseisen asiakkaan tietojen generointiin lomakkeelle samoin kuin saman osion asiakastietojen muokkaukseen tarkoitetussa toiminnossa. Lisäksi asiakkaan poiston on toimittava myös tässä tapauksessa.

3. Varmista, että kun hakukenttään liittyvä valintaruutu klikataan aktiiviseksi, hakutoiminto listaa vain vuosihuoltosopimukset ja kun sen poistaa käytöstä, toiminto ei tee kyseistä rajausta.

Kaikki testattu ja tarvittavat korjaukset tehty 18.8.2020.

## SELAA TOIMENPITEITÄ-OSIO

1. Klikkaa Selaa toimenpiteitä-osiota. Varmista, että pudotusvalikossa on valittuna ”Kaikki” ja että events-tietokantataulun kaikki tapahtumat ovat riveittäin näkymässä. Jokaisella tapahtumalla on siis oma rivinsä.

1.1 Kullakin rivillä tulee olla oma Muokkaa- ja Poista-painikkeensa. Muokkaa-painike tulee näkyä sinipohjaisena ja Poista-painike punapohjaisena.

1.2 Varmista, että kullakin rivillä näkyy toimenpiteen aloitusajankohta, sen tyyppi, asiakkaan nimi, toimitusosoite ja paikkakunta (jos kaikki kyseiset attribuutit ovat lisättynä tapahtumalle).

1.3 Varmista, että huoltokäynnit näkyvät listassa punapohjaisena, vuosihuoltokäynnit oranssipohjaisena, vuosihuoltosopimuskäynnit keltapohjaisena, valmiiksi kuitatut vuosihuoltosopimuskäynnit vihreäpohjaisena, lähetä tuote ja soittopyynnöt sinipohjaisena.

1.4 Varmista, että valitsemalla pudotusvalikosta ”Joonas” vain hänelle osoitetut toimenpiteen tulostuvat vastaavasti kuin kohdissa 1.2 ja 1.3.

1.5 Varmista, että valitsemalla pudotusvalikosta ”Vellu” vain hänelle osoitetut toimenpiteen tulostuvat vastaavasti kuin kohdissa 1.2 ja 1.3.

1.6 Varmista, että kun Muokkaa painiketta klikataan, toimenpiteen tiedot generoidaan vastaavasti kuin sitä vastaavassa events-tietokantataulun tietueessa mukaan lukien tiedostot.

1.7 Toista kohta 1.6 eri toimenpiteen kohdalta ja varmista, ettei edellisen toimenpiteen tietoja jää lomakkeelle, vaan että lomake täytetään juuri valitun toimenpiteen tiedoilla.

1.8 Varmista, että toimenpiderivin Poista-painike toimii niin, että se poistaa toimenpiteen (tietokannasta) sekä siihen liittyvän tiedoston, jos toimenpiteeseen liittyvän asiakkaan tiedoissa ei ole samaa tiedostoa tai jos samaa tiedostoa ei ole muilla vastaavilla tietorakenteilla – muussa tapauksessa tiedoston tulee jäädä files-kansioon.

1.9 Varmista toimenpiteiden muokkaustoiminnon oikeellisuus vastaavasti kuin Asiakasrekisteri-osion toimenpiteen muokkaustoiminnossa.

Kaikki testattu ja tarvittavat korjaukset tehty 22.8.2020.

**Hakukentän testaus**

1. Toista Asiakasrekisteri-osion hakukentän testaukseen liittyvät testit niin, että tässä osiossa hakukohteena ovat toimenpiteet – ei itse asiakkaat, kuten asiakasrekisteri-osiossa. Huom! Hakukentän on toimittava riippumatta pudotusvalikon asennosta

Kaikki testattu ja tarvittavat korjaukset tehty 24.8.2020.

# KALENTERI-OSIO

1. Varmista, että kummankin työntekijän toimenpiteet tulostuvat oikein kalenterinäkymään (sijoitus datetime-arvon perusteella ja niiden taustavärit vastaavat kullekin toimenpiteelle varattua – valmiiksi kuitattu toimenpide tulee näkyä vihreäpohjaisena).

2. Varmista, että kalenterin kautta avautuvan lightbox-ikkunan kautta on mahdollista muokata toimenpiteen ajankohtaa. Tarkista muutosten oikeellisuus tietokannasta ja kalenterinäkymästä.

3. Varmista, että lightbox-ikkunan Avaa lomakkeessa-painikkeen kautta avataan toimenpide Selaa toimenpiteitä-osion lomakkeella.

3.1 Tee vastaavat varmistukset kun Selaa toimenpiteitä-osion Muokkaa-painikkeen käytön yhteydessä eli että lomakkeelle generoidaan ko. toimenpiteen tiedot tiedostoineen oikein.

3.1.1 Varmista myös, että muokkaukset tallentuvat vastaavasti kuin Selaa toimenpiteitä-osion vastaavassa tilanteessa.

4. Varmista, että toimenpiteen perustiedot näytetään ruudulla, kun hiiren kursori viedään toimenpiteen kohdalle.

5. Varmista, että toimenpidettä voidaan siirtää myös raahamalla (drag and drop).

Kaikki testattu ja tarvittavat korjaukset tehty 27.8.2020.

# ASETUKSET-OSIO

1. Täytä kukin input-kenttä jollakin merkkijonolla ja tarkista, että tekstiviestipalvelun tunnus ja salasana sekä työnhallintajärjestelmän käyttäjätunnus ja salasana päätyvät asetukset-tietokantatauluun vastaavasti.

1.1 Varmista, että kenttien ”Puhelinnumero Vellu” ja ”Puhelinnumero Joonas” syötteet päätyvät tyontekijat-tauluun vastaavasti.

2. Varmista, että asetukset- ja tyontekijat-tietokantataulujen data generoidaan sivun syötekenttiin Asetukset-valikkovalinnan valinnan jälkeen.

Kaikki testattu ja tarvittavat korjaukset tehty 27.8.2020.