

Opinnäytetyö (YAMK)

Teknologiaosaamisen johtaminen

2021

Juuso Järvinen

SPA-SOVELLUKSEN TOTEUTTAMINEN HAASTATTELUTUTKIMUKSEN AVULLA

Juuso Järvinen

SPA-SOVELLUKSEN TOTEUTTAMINEN HAASTATTELUTUTKIMUKSEN AVULLA

Tämä opinnäytetyö käsittelee projektityöskentelyä ohjelmistoprojektissa yleisellä tasolla. Tavoitteena oli toteuttaa yritykselle uusi tapahtumahallinnan ohjelma. Yritys halusi uudistaa nykyistä ohjelmaansa ja toteuttaa uuden ohjelman nykyaikaisilla ohjelmistoalustoilla. Opinnäytetyön ensisijainen tavoite oli tutkia, miten ryhmähaastattelun avulla pystytään helpottamaan ohjelmistoprojektin suunnittelua ja toteuttamista.

Työn määrittelyt tehtiin ryhmähaastattelun avulla ja kehittäminen toteutettiin Angular-sovelluskehityksen avulla. Ryhmähaastattelun avulla haluttiin antaa jokaiselle asiakkaalle mahdollisuus vaikuttaa seuraavaan ohjelmaversioon. Haastattelun kysymykset pidettiin mahdollisimman avoimina, jotta syntyisi paljon keskustelua. Keskustelun avulla pyrittiin saamaan aikaiseksi kaikkia palveleva kokonaisuus.

Haastattelun avulla saatujen määritysten suuren määrän vuoksi oli hankala löytää kaikkia palvelevat ominaisuudet ensimmäiseen ohjelmaversioon. Projektissa käytettiin hyväksi ketterää kehitysmenetelmää, jonka avulla asiakas pidettiin mukana koko projektin ajan.

Projektin toteutus onnistui hyvin. Ryhmähaastatteluja pidettiin toimivina asiakkaan ja projektitiimin toimesta. Kyseinen toimintamalli loi tiiviin tavan toimia monen asiakkaan kanssa projektin läpi. Moni haastatteluun osallistunut pohtii kokeilevansa samaa kehitystapaa omissa projekteissaan.

Tästä opinnäytetyöstä saadaan uusia näkökulmia ohjelmistoprojektin kehittämiseen. Työssä kerrotaan yleisesti IT-projektinhallinnasta sekä projektin vaiheista ja päättämisestä. Opinnäytetyössä käydään läpi web-kehityksen eri menetelmiä ja käytäntöjä.

ASIASANAT:

Projekti, projektikehitys, web-kehitys, ryhmähaastattelu, ketteräkehitysmenetelmä

MASTER'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Master's Degree Programme in Technological Competence Management

2021 | 48 pages

Juuso Järvinen

IMPLEMENTING THE SINGLE PAGE APPLICATION USING AN INTERVIEW RESEARCH

The present Master's thesis discusses working with software projects on the basic level. The aim is to plan a new event management software for enterprise. The company wanted to renew their existing software and implement a new one with modern software platforms. The primary goal of the thesis is to study how using group interviews improves the planning and implementing of a software project.

The specifications of the project are done with a group interview and developing is executed by using the Angular application framework. With the help of the group interview, it is possible to give each customer an opportunity to influence the next program version. The questions of the interview were kept as open as possible to create a lot of discussion. The aim of encouraging discussion was to create an entirety that would serve each participant.

Due to the large number of software specifications obtained through the interview, it was difficult to find all the features that would meet the needs of each participant in the first version. An agile development method was used in the project to keep the client involved throughout the project.

The implementation of the project was successful. Both the client and the project team considered the group interviews a successful way of working. The approach created a close-knit way of working with the clients throughout the project. Many of the clients that were interviewed considered trying out the same development method in their own projects.

The present thesis provides new perspectives on software project development as well as general information on IT project management, phases and completion. It also reviews the various methods and practices of web development.

KEYWORDS:

Project, project development, web development, group interview, agile development method

SISÄLTÖ

KÄYTETYT LYHENTEET TAI SANASTO	6
1 JOHDANTO	8
2 WEB-SOVELLUSKEHITYKSEN ERI MENETELMÄT	10
2.1 Monisivuinen web-sovellus	10
2.2 Yksisivuinen web-sovellus	10
2.3 Sovelluksen arkkitehtuuri	11
2.4 Sovelluskehysten tärkeys	13
2.5 Web-komponentit	14
2.6 Web-komponentit standardiin kuuluvat teknologiat	14
2.7 Räätylöidyt elementit	16
3 PROJEKTIHALLINTA	18
3.1 Perinteinen projektimalli (vesiputousmalli)	19
3.2 Ketterä projektimalli	20
3.3 Projektin epäonnistumiseen vaikuttavat tekijät	22
3.4 Suunnittelu	23
3.5 Projektin vaatimusmäärittelyt ja elinkaari	25
3.6 Toteutus	27
3.7 Testaus	28
3.8 Projektin päättäminen	28
3.9 Projektin jatkokehitys	29
4 RYHMÄHAASTATTELU	30
4.1 Tutkimusongelma	30
4.2 Haastattelututkimus	31
4.3 Haastattelun toteutus	32
4.4 Haastattelun tulokset	33
5 VAATIMUSMÄÄRITTELY JA TOTEUTUS	36
5.1 Nykyinen koulutuksenhallinohjelmisto	36
5.2 Kehittämiskohteiden valinta	36
5.3 Käyttöliittymäsuunnittelu	38
5.4 Rajapinnan suunnittelu	40

5.5 Toteutus ja testaus	40
5.6 Palaute ja kehitysehdotukset	41
6 POHDINTA JA JOHTOPÄÄTÖKSET	42
7 YHTEENVETO	45
LÄHTEET	46

KÄYTETYT LYHENTEET TAI SANASTO

Ajax	JavaScriptin ja XML:n tekniikka, joka mahdollistaa verkkosivujen päivittämisen asynkronisesti (Asynchronous JavaScript And XML)
Angular	Googlen kehittämä JavaScript pohjainen sovelluskehys
ASP.NET	Microsoftin kehittämä avoimen lähdekoodin kehitysalusta monenlaisten sovellusten rakentamiseen.
DOM	Dokumenttioliomalli, jolla kuvataan HTML:n rakenne (Document Object Model)
HTML	HTML on merkintäkieli, jota tarkastellaan verkkoselaimessa. HTML-tiedosto koostuu tekstistä, jossa tiedoston rakenne merkitään elementeillä. Elementit merkitään tagien väliin. Elementeillä on aina aloitus- ja lopetustagi, joiden välissä on varsinainen elementin sisältö. (Hypertext Markup Language)
HTTP	Protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon (Hypertext Transfer Protocol)
JavaScript	Oliopohjainen ohjelmointikieli, jota käytetään pääsääntöisesti web-kehityksessä
JAVAFX	Sovelluskehys työpöytä, mobiili ja sulautetuille järjestelmille
JSON	Avoimen standardin kevyt tiedostomuoto tiedonvälitykseen (JavaScript Object Notation)
MVC	Suunnittelumalli, jonka tärkein tehtävä on eriyttää sovelluksen esitys- ja logiikkakerros toisistaan ja delegoida sovelluksen sisäisiä vastuita eri osa-alueille. (model-view-controller)
MVVM	Ohjelmistoarkkitehtuurimalli, joka helpottaa graafisen käyttöliittymän kehityksen erottamista (Model-View-ViewModel)
Ruby on Rails	avoimen lähdekoodin verkkosovelluskehys, joka pohjautuu MVC-arkkitehtuuriin

SPA	Singe-page app on sovellus, jossa kaikki ladataan kerralla selaimeen, mutta kaikkea ei näytetä kerralla. Pystytään luomaan nopeampi sovelluksia (single-page application)
Spring	Java pohjainen verkkosovelluskehys
Typescript	Avoimen lähdekoodin ohjelmointikieli, joka perustuu JavaScriptiin

1 JOHDANTO

Selainpohjaisten ohjelmistojen merkitys työskentelyssä kasvaa jatkuvasti. Samalla käyttäjien vaatimukset ja tietoisuus ohjelmista lisääntyvät jatkuvasti. Tämän vuoksi ohjelmistotalalla kilpaillaan tilaajista ja pyritään pitämään markkinajohtajuus oman alan ohjelmilla. Ohjelmistojen elinkaari tulee päätökseen, joten yrityksen tulee uudella sovelluksella tehostaa ja helpottaa tilaajan työntekoa.

Web-ohjelmistojen tekniikat ja toteutustavat ovat kehittyneet vuosien varrella. Nykyisillä tekniikoilla pystytään luomaan ohjelma, joka toimii eri päätelaitteilla ja selaimilla. Yhden sivun verkkosovellukset (SPA, Single Page Application) ovat mahdollistaneet nopean kehityksen alalla. SPA-sovelluksessa voidaan poistaa ylimääräiset sivulataukset ja toteuttaa visuaalisempia kokonaisuuksia. Kokkonen on tutkimuksensa tuloksena todennut, että Single Page Application -sovelluskehikset ovat useissa tilanteissa käyttökelpoisia yritysohjelmistojen kehityksessä. (Kokkonen 2015) Käyttöliittymästä voidaan luoda käyttäjää ohjaava ja selkeästi käytettävä ohjelma. SPA-sovellukset hajautettuna järjestelmänä -maisteritutkielmassa kirjoitetaan, kuinka SPA-sovelluksesta pystytään luomaan työpöytäsovelluksen kaltainen. (Elorannan 2020)

Tässä opinnäytetyössä suunnitellaan ja toteutetaan yrityksen sekä sen asiakkaiden kanssa yhteistyössä nykyaikainen ja interaktiivinen web-sovellus. Kyseisessä opinnäytetyössä suunnitellaan ja toteutetaan SPA-sovellus, joka tehdään JavaScript pohjaisen sovelluskehiksen alustalle (Angular 2). Ohjelman suunnittelu ja määrittely tehdään useamman asiakkaan kanssa yhdessä. Yritys on SaaS-ohjelmistoyritys. Yrityksen toiminta perustuu pitkälle vietyyn tuotteistukseen ja valmiisiin modulaarisiin tuotteisiin. Yritys tarjoaa eri kokoisille yrityksille osaamisen ja koulutusten hallintaratkaisuja, sekä lehtitilausten hallintaa ja ratkaisua suojattujen sähköpostien ja tiedostojen lähettämiseen. Yrityksen vahvuutena on nopeat käyttöönotot ja asiakaskohtaisesti muokattava moduulit. Tässä opinnäytetyössä tavoitteena on kehittää nykyaikaisempi, käytettävämpi ja ylläpidettävämpi ohjelma.

Kehittämiprojekti toteutetaan ketterää projektisuunnitelmaa käyttäen. Ketterä projektisuunnitelma soveltuu parhaiten tämän kehittämiprojektin toteuttamiseen, koska tarkoituksena on toteuttaa projekti asiakkaiden kanssa yhteistyössä joustavasti ja nopeasti. Asiakkaille järjestetään haastattelutilausuus, josta pyritään saamaan määrittelyt uutta ohjelmaa varten.

Haastattelututkimuksen avulla tavoitteena on luoda kaikkia palveleva sovellus, jota on jatkossa helppo kehittää ja samalla pidentää ohjelman elinkaarta. Haasteena projektissa pidetään sitä, miten asiakkaat saavat yhdessä määriteltä toimivan ohjelman. Tutkimuksella halutaan tuoda erilainen näkemys projektin määrittelyyn. Jokainen asiakas pystyy tuomaan ja kertomaan omia tarpeitaan muille käyttäjille.

Luvussa 2 käydään läpi web-sovelluksen toimintaperiaatteet. Lukijalle avataan näkyvyys SPA-sovelluksen kehittämisestä ja sen toiminnoista. Luvussa 3 käsitellään IT-projektin periaatteet ja sen kehityskaari sekä tuodaan esille, miten luodaan onnistunut projekti. Tutkimuksen toteutus on seuraavan luvun 4 aiheena, siinä esitellään tutkimusongelma ja haastattelututkimuksen toteutusta. Viidennessä luvussa suunnitellaan ja toteutetaan projekti. Johtopäätöksissä käydään läpi tutkimuksen tulokset ja projektin onnistuminen.

2 WEB-SOVELLUSKEHITYKSEN ERI MENETELMÄT

2.1 Monisivuinen web-sovellus

Perinteisimmissä web-sovelluksissa käyttäjälle ladataan sivu esiin palvelinpyyntöjen avulla. Tämä näkyy usein käyttäjille sivun välkkymisenä, koska sivu ladataan uudestaan halutuilla tiedoilla. Sivun lähettämiseen palvelimelle hypertekstin siirtoprotokollapyynnön (HTTP, Hypertext Transfer Protocol), johon palvelin vastaa lähettämällä hyperlinkkejä sisältävää tekstiä (HTML-sivun, Hypertext Markup Language). HTTP-pyyntöissä käytetään hyväksi eri tyyliä metodeita, jotka määrittelevät kutsun tyyppin. Yleisimmät tyypit ovat GET, POST, PUT ja DELETE. GET-metodin avulla käyttäjälle haetaan esimerkeiksi verkkosivu ja muut tiedot. POST-metodin avulla palvelimelle voidaan lähettää esimerkiksi täytetyn lomakkeen tiedot ja tallentaa ne tietokantaa. PUT-metodin tarkoituksena tallentaa sivu. DELETE-metodia käytetään sivun poistamiseen.

Monisivuinen web-sovellus toimii usein hitaammin kuin yksisivuiset ohjelmat, koska se joutuu aina lataamaan sivun uudelleen. Tämä vaikuttaa käyttäjäkokemukseen usein negatiivisena asiana. Ohjelman logiikka, resurssit ja data sijaitsevat palvelimella.

2.2 Yksisivuinen web-sovellus

Yksisivuisessa web-sovelluksessa on vain yksi sivu, jossa ladataan HTML-dokumentti kerran ja tämän jälkeen vain sisältöä muutetaan. Yksisivuinen sovellus lähettää ensimmäisellä kerralla HTTP pyynnön palvelimelle ja tämän jälkeen pyynnot tehdään Ajax-kutsuina ja palvelin palauttaa vastaukset JSON-tyyppisenä. Yksisivuinen web-sovellus on käyttäjälle lähes yhtä helppo ja interaktiivinen käyttökokemus kuin normaali työpöytäsovellus. Yksisivuinen web-sovellus toimii myös yhteydettömässä (offline) tilassa, koska sovelluksen tila on tallennettu selaimen muistiin. Selaimen sulkemisen jälkeenkin tiedot löytyvät muistista (Fink & Flatow 2014). Yksisivuisen sovelluksen logiikka toimii pääsääntöisesti selaimessa, on sen käyttäminen erittäin nopeaa. Validoinnit, tietokantakyselyt ja autentikoinnit tapahtuvat ainoastaan palvelinpäässä ensimmäisellä kerralla. SPA-ohjelmilla on monia etuja, kuten parantunut sovellusten suorituskyky ja yhdenmukaisuus sekä pienemmät kehitys- ja infrastruktuurikustannukset. (Lawson 2018)

2.3 Sovelluksen arkkitehtuuri

Angular on Googlen suunnittelema avoimen lähdekoodin sovelluskehys. Sovelluskehys on suunniteltu vaivattomampaan, nopeampaan ja testaavampaan kehittämiseen. Angular-sovellukset toimivat jokaisella päätelaitteella. Angularilla voidaan tehdä web-sovelluksia tai natiivisovelluksia esimerkiksi puhelimen sovelluskauppaan. Angular-sovellukset ovat moduuleista koottuja kokonaisuuksia. Angularin arkkitehtuuri ei varsinaisesti koostu mistään standartista suunnittelumallista, vaan se mahdollistaa useamman suunnittelumallin käytön. Angularin yhteensopivuus useamman arkkitehtuurimallin kanssa avaa monia mahdollisuuksia kehittäjälle. Angularissa yhdistetään monia pieniä komponentteja, joiden avulla komponentit kasvavat valmiiksi sovellukseksi (Chandermani 2018). Angular-kehitys tapahtuu JavaScript- ja TypeScript-sovelluksissa kirjastoa hyväksi käyttäen.

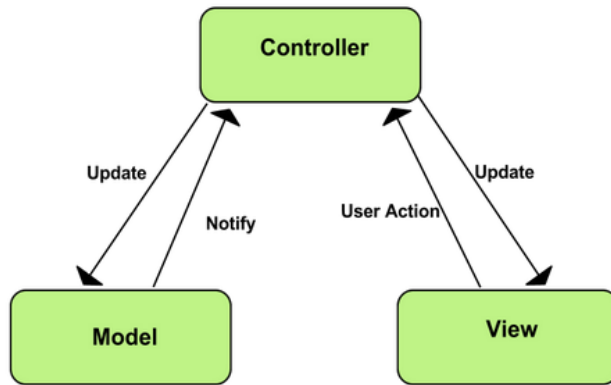
Spring on Java-sovelluskehys, jota käytetään erityisesti yritysohjelmistoissa. Spring-sovelluskehysten avulla pyritään helpottamaan Java-sovellusten kehittämistä. Spring on modulaarinen ohjelmointikieli, joka kehittyy jatkuvasti ja sillä kehitetään natiivisovelluksia Windows- ja Mac-tietokoneille.

ASP.NET on Microsoftin kehittämä sovelluskehys. Sen avulla voidaan rakentaa dynaamisia web-sovelluksia, natiivisovelluksia tai web-palveluja. Kyseisellä sovelluskehyksellä toteutetaan usein raskaita ja suuria ohjelmia. Ohjelmointi tapahtuu serverikoodin puolella, joka palauttaa arvot käyttöliittymälle.

Ruby on Rails on avoimen lähdekoodin ohjelmistokehys, jota käytetään web-sovellusten tekemiseen. Rubyssa ohjelmakoodin määrä jää usein pienemmäksi kuin muilla ja kehys tuottaa ominaisuuksia myös automaattisesti. Ohjelmointi tapahtuu serverin puolella, joka usein näkyy käyttäjälle pieninä viiveinä ohjelmassa.

Perinteisemmässä mallissa on monia eri arkkitehtuurimalleja, joista yleisimpiä ovat MVC (Model-View-Controller), Microsoftin kehittämä MVVM-mallin arkkitehtuuri (Model, View, ViewModel) ja Facebookin kehittämän Flux-arkkitehtuurimalli. MVC-arkkitehtuuri on peräisin jo 1970-luvulta ja malli on vieläkin laajassa käytössä. MVC on todella suosittu erityisesti web-sovelluksissa, joissa käydään vuorovaikutusta käyttäjän kanssa. Tämä näkyy käyttäjälle siten, että aina kun hän painaa jotain, latautuu uusi virtaus esiin. MVC-mallissa dataa voidaan luoda, hallita ja muokata. Spring, ASP.NET ja Ruby on Rails

hyödyntävät usein MVC-arkkitehtuurimallia. Arkkitehtuurissa näkymä ja ohjain ovat riippuvaisia mallista, mutta malli ei ole riippuvainen ohjaimesta tai näkymästä. Ohjain toimii välittäjänä, joka vie mallilta dataa näkymälle. Kuvassa 1 ohjain päivittää mallia ja näkymään ja käyttäjän tekeminen vaikuttaa ohjaimeen (developer.chrome 2014)

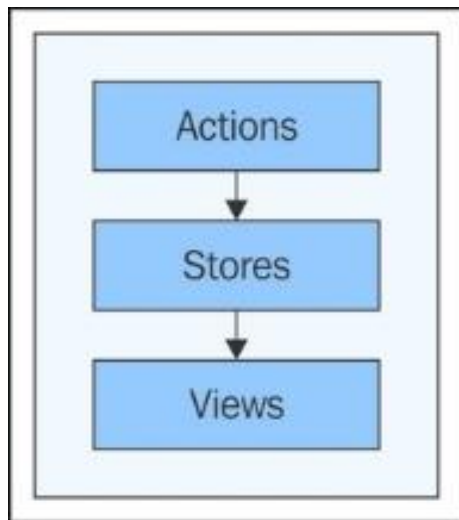


Kuva 1. MVC-arkkitehtuurin kuvaus (developer.chrome 2014)

MVVM-mallin avulla pyritään helpottamaan ohjelmointia modulaarisuudella. Mallin arkkitehtuuri on yksinkertaista, siinä on hyvät työkalut ja se on helppo toteuttaa. MVVM-mallin avulla käyttöliittymä pystytään erottamaan muusta ohjelmistosta, mikä helpottaa ylläpitämistä. Kyseessä on vielä tuore malli, joten sitä ei ole sidottu useaan ohjelmointikieleen automaattisesti. JavaFX avulla voidaan aloittaa suoraan käyttämään MVVM-mallin arkkitehtuuria. JavaFX avulla voidaan kehittää asiakasohjelmistoalusta työpöydälle, mobiililaitteille ja sulautetuille järjestelmille. SPA-malli mahdollistaa työpöytäsovellusten kaltaisen kokemuksen web-selaimessa, sillä jokaista sivuvaihtoa varten ei jouduta tekemään pyyntöä palvelimelle, vaan näkymät muodostetaan selaimessa paikallisesti etukäteen ladatun datan avulla. (Eloranta 2020)

Flux-sovelluksilla on neljä pääosaa: välittäjä, kauppa, näkymä ja luettu. Flux on avoimen lähdekoodin ohjelmistokehys. Yksi Flux-suunnittelijoiden tekemä päätös erottaa se muista; Flux-arkkitehtuurit eivät välitä siitä, kuinka käyttöliittymäelementit esitetään. (Boduch 2016). Fluxissa näkymäosa hakee kaupasta aina uutta tietoa. Facebookissa tämä näkyy käyttäjille uusina postauksina, joita ilmestyy jatkuvasti ruudulle. Tämän avulla voidaan luoda sovellus, joka toimii datavirtana ja käyttäjä välttää tekemästä useampaa kertaa saman postauksensa. Fluxin käyttöä suositellaan sovelluksissa, joissa

tieto näytetään virtana. Web-pohjaisessa sovelluksessa logiikasta tulee helposti liian haastavaa ylläpitää. Kuvassa 2 kuvataan, miten tieto liikkuu virtauksena. (Boduch 2016)



Kuva 2. Flux-arkkitehtuurin kuvaus

2.4 Sovelluskehysten tärkeys

Angular kuuluu JavaScript-pohjaisiin kehyksiin, jonka avulla pyritään helpottamaan web-ohjelmointia. Tässä opinnäytetyössä käytetään JavaScriptistä kehittyneempää versiota eli TypeScriptiä. Kyseinen kehys antaa käyttäjälle mahdollisuuden muokata suoraan HTML-DOMEja. Kehittämisen helpottamiseksi löytyy paljon valmiita kirjastoja ja komponenttikokonaisuuksia.

Web-sovelluksia voidaan kehittää nykyään JavaScriptillä myös ilman sovelluskehystä (Fain & Moiseev 2015). Tämä antaa käyttäjälle paljon vapauksia kehittää sovellusta, mutta laajempien kokonaisuuksien ylläpito käy usein hankalaksi. Datat hallinta on usein suurin haaste sovelluskehuksettömässä kehittämisessä ja jos sovellus optimoidaan toimimaan useammalla eri selaimella (Fain & Moiseev 2015), törmäävät kehittäjät edellä mainittuihin ongelmiin.

2.5 Web-komponentit

Web-komponenttien avulla voidaan kehittää itsenäisiä sovellusliittymiä tai sovelluksen osia. Ne ovat uudelleenkäytettäviä ja mukautettuja elementtejä, ja komponentit ovat kapseloituja pois muusta koodista. Komponenttien avulla koodi pysyy selkeänä ja yksinkertaisena. Web komponentteja kutsutaan tuttavallisemmin pienisohjelmaksi (widget) ja ne toimivat nykyaikaisissa selaimissa, minkä tahansa JavaScript-kirjaston tai sovelluskehiksen kanssa. (webcomponents.org 2021) Jokainen komponentti on uudelleenkäytettävissä ohjelmassa niin monta kertaa kuin tarvitsee. Web-komponentit -arkkitehtuuri mahdollistaa web-sovelluksen uudelleenkäytön ja kapseloinnin, mikä helpottaa generistien sovellusten kehittämistä. Web-komponentit luodaan HTML-, CSS ja JavaScript teknologioita hyväksikäyttäen, koska lähes jokainen selain tukee näitä.

Angular 2 projekteihin sisällytetään useita web-komponentteja. (Wilken 2018) Angularin omat komponentit ovat hyvin samankaltaisia, kuin web-komponentit. Angular 2 käyttää selaimissa hyväksi web-komponenttien toimintaan suunniteltuja ominaisuuksia. Angularille on rakennettu jatkuvasti kasvava komponenttikirjasto. (Wilken 2018)

2.6 Web-komponentit -standardiin kuuluvat teknologiat

Komponentit ovat kehityksessä vanha unelma, jossa voi tarttua valmiiksi toimiviin toimintoihin ja tuoda ne omaan sovellukseensa. (Exbrayat 2016) Web-komponentit ovat joukko-liittymiä, joista muodostuu verkkosovelluksia ja verkkosivuja. Web-komponenttien standardit muodostuvat olemassa web-standardeista. (web-components, 2021) Web-komponentit jaetaan neljään eri päämäärittelykseen:

- Custom elements, jossa voidaan suunnitella ja luoda uudentyyppisiä DOM-elementtejä
- Shadow DOM, joka määrittelee komponenttien tyylit ja voi kapseloida tyylit komponentiksi.
- HTM-malli kokoaa elementit yhteen ja näyttää käyttäjälle.
- ES-moduulit määrittelee moduulien uudelleenkäyttöä.

Jokaista web-komponentti -standardia voidaan käyttää myös itsenäisesti. Kaikki selaimet eivät vielä tue kaikkia standardeja täydellisesti. Chrome ja Firefox antavat tällä hetkellä parhaan tuen standardeille.

2.7 Räätelöidyt elementit

Yksi web-komponentti -standardin tärkeimmistä ominaisuuksista on kyky luoda omia HTML-elementtejä eli räätälöity elementti (Custom elements). Mukautetut elementit ovat joukko sovellusliittymiä, jotka tarjoavat kehittäjille tavan laajentaa HTML-elementtejä, rakentaa uusia ja määrittää niiden käyttäytymisen (The Ultimate Guide to Web Components 2019). HTML-elementti voidaan luoda *esimerkiksi* `<custom-komponentti></custom-komponentti>`, joka sisältää kapseloidut toiminnallisuudet ja muotoilut. Custom elements -standardin komponentille voidaan määritellä esimerkiksi oma käyttöliittymä. Luotu elementti voidaan rekisteröidä DOM:iin helposti JavaScriptin avulla. (Exbrayat 2016)

Angular 2 web-sovelluksissa hyödynnetään usein Custom elements komponentteja. Tämän mahdollistaa omien komponenttien käytön sovelluksessa. Samaa komponenttia voidaan käyttää useampaan kertaan eri kohdassa sovellusta. Komponentti lisätään esimerkiksi toisen komponentin sisään HTML-tagina `<custom-komponentti></custom-komponentti>`

Varjo-DOM

Varjo-DOM-teknologia (shadow DOM) mahdollista elementin kapseloinnin muusta sovelluksesta erilleen. Kyseinen DOM elementti kuuluu osaksi koko sovelluksen DOM:a, mutta se on suojattu pääsovelluksen CSS-määrittelyiltä ja DOM-manipulaatiolta. Komponentti toimii missä tahansa ympäristössä, vaikka sivulla olisi toista CSS:ää tai JavaScriptiä (The Ultimate Guide to Web Components 2019).

Angular 2 -sovelluksissa Shadow DOM -teknologia mahdollistaa komponenttien itsenäisen toiminnan. Sovellukset koostuvat tyypillisesti monesta komponentista, joten on tärkeää estää niitä vaikuttamasta ei-toivotulla tavalla toisiinsa.

HTML-mallit

HTML-mallien (HTML templates) avulla luodulle elementille määritellään sivu pohja. HTML-malleja ei renderöidä sivun lataamisen yhteydessä, mutta jotka voidaan kloonata ja lisätä asiakirjaan ajon aikana JavaScriptin avulla (The Ultimate Guide to Web

Components 2019). Sivupohja luodaan *<template>*-elementin avulla. Template-sivupohja voidaan ottaa käyttöön vasta kun sitä tarvitaan, eli se voidaan kapseloida erilleen muusta dokumentista.

HTML-teknologiaa hyödynnetään Angular-sovelluksissa lisäämällä sivupohja osaksi komponenttia. Templaten avulla jokaiselle komponentille voidaan määritellä omanlainen ulkoasu.

HTML-tuonti

HTML-tuonti (HTML imports) on tarkoitettu verkkokomponenttien pakkausmekanismiksi, mutta voidaan käyttää HTML-tuontia (Mozilla Developer Network 2020). Teknologia mahdollistaa myös HTML-dokumenttien tuonnin ja uudelleen käyttämisen toisissa HTML-dokumenteissa.

HTML imports mahdollistaa valmiiden komponenttien käytön sovelluksessa. Kyseisiä komponentteja on saatavilla monissa eri Angular 2 kirjastoissa. Tämä nopeuttaa huomattavasti Angular 2 sovelluksen kehittämistä.

3 PROJEKTIHALLINTA

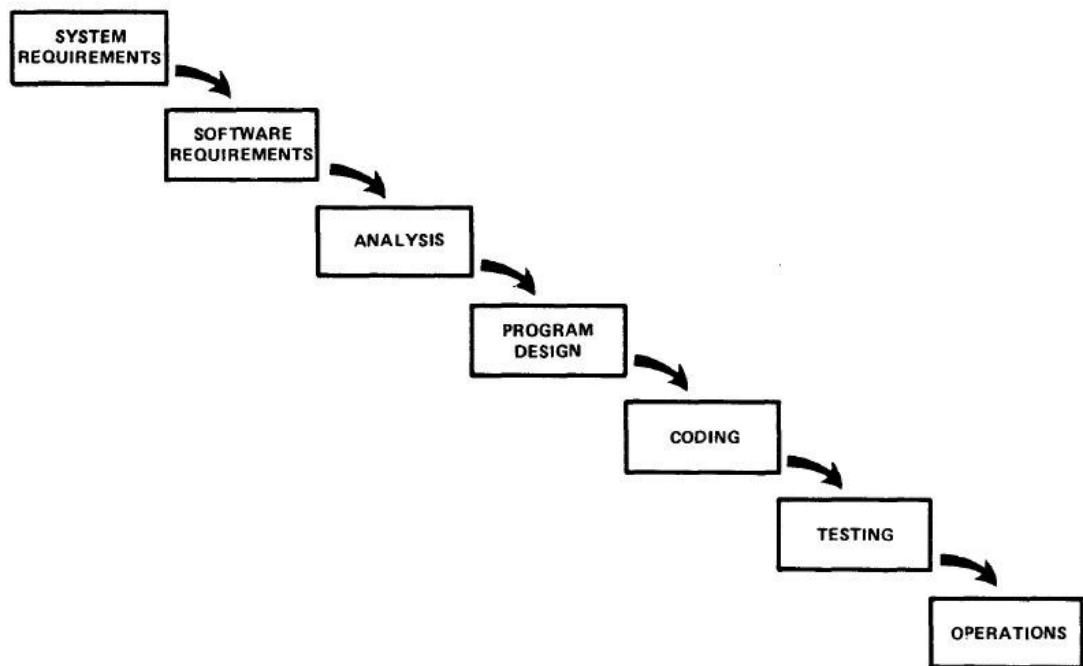
Projektien avulla pyritään organisoitumaan ja parantamaan ohjattavuutta. Projektilla on aina alku ja loppu ja jokaiselle resurssille on määritelty oma paikkansa. Usein projektien tavoitteena on tehostaa työskentelyä, pienentää kuluja ja tuottaa parasta laatua. Ohjelmistoprojekteissa pyritään luomaan käyttäjäystävällinen ja selkeä kokonaisuus, sekä samalla huomioimaan jatkokehitysmahdollisuudet ja mahdollisimman pitkä elinkaari. Hyvän IT-projektin tunnistaa siitä, että se ei ole IT-projekti ollenkaan. IT-projektit eroavat jonkin verran muista projekteista ja tässä opinnäytetyössä hyödynnetään IT-projektinhallintaa. Onnistuneen IT-hankkeen tavoitteet eivät liity IT:hen vaan siihen, että teknologian avulla saavutetaan selkeästi määritellyt liiketoiminnalliset päämäärät. (Korsström, 2013) IT-projekti vaiheistetaan tyypillisesti neljään vaiheeseen:

1. Projektin aloittaminen
2. Projektin suunnittelu
3. Projektin seuranta ja ohjaus
4. Projektin päättäminen

Tyypillisesti IT-projektin päävastuu on projektipäälliköllä, jonka tehtävänä on saavuttaa onnistunut projekti sille asetetuilla tavoitteilla. Projektipäällikkö muokkaa ja ohjaa resursseja projektin läpi. Samalla sen tulee pitää projektinohjausryhmä tietoisena aikataulusta ja etenemisestä. Tutkimusten ja IT-projektien perusteella nousee esiin tiettyjä asioita, joita tulee huomioida IT-projekteissa. Elo (2014) painottaa viestinnän merkitystä IT-projekteissa, jotta projekti saadaan onnistumaan. IT-projektin onnistumisen määrittely on nykypäivänä hankalaa. IT-projektin onnistumisen arvioiminen on vaikeaa, koska onnistumiskriteerit voivat vaihdella tilaajan ja tekijän välillä. Projektinhallinnallisia ongelmia on helppo tunnistaa alkaen määrittelyjen heikkoudesta, osapuolten sitoutumattomuudesta, projektin seurannasta sekä teknisistä asioista. (Vainikainen 2019) Määritelmien mukaisesti ja aikataulussa toteutunutta projektia voidaan kuitenkin pitää onnistuneena. Laitisen (2020) mukaan projektien onnistumiseen IT-alalla vaikuttavat onnistumiskriteerit. Onnistuneen projektin tarkastelunäkökulmana voivat olla esimerkiksi rahoittajan, kehittäjän, projektipäällikön, asiakkaan, käyttäjän tai muun sidosryhmän näkökulma. Vaihteleva näkemys ja vaihteleva onnistuneen projektin määritelmä luovat haasteita, kun vertaillaan keskenään onnistumis- ja epäonnistumistekijöitä.

3.1 Perinteinen projektimalli (vesiputousmalli)

Vesiputousmallia pidetään usein perinteisenä projektikehitysmallina. Vesiputousmallissa projekti etenee yksi askel kerrallaan. Kyseistä mallia käytettiin pitkään projekteissa ennen kuin ketterät projektimallit otettiin käyttöön. Vesiputousmallin avulla tilaajan tietää, mitä alun perin on tilannut ja samalla projektille saadaan selkeät arviot aikatauluista ja budjetista. Projektin seuraaminen on helppoa projektiryhmälle, koska koko ajan tiedetään missä mennään ja mikä vaihe tulee seuraavaksi. Kyseistä tapaa voidaan käyttää suuressakin organisaatiossa, mikäli dokumentaatio on kunnossa. Kuvassa 3 on kuva vesiputousmallista, jossa näkyy projektin eteneminen seuraavaan vaiheeseen ja jossa edelliseen osaan ei voida palata (Lekman 2010).



Kuva 3. Vesiputousmalli (Lekman 2010)

Vesiputousmallista löytyy myös useampi heikkous, jotka voivat johtaa projektin epäonnistumiseen. Asiakas näkee tuotteen vasta valmiina, joten käytettävyyssongelmat ja muut virheet paljastuvat vasta projektin lopussa. Valmiin projektin korjaaminen ja muuttaminen voi tulla kalliimmaksi kuin alkuperäinen projekti oli. Vesiputousmallilla työskentely on joustamatonta ja ei anna mahdollisuutta virheille. Jokaisen vaiheen jälkeen tarkistetaan projektin suunniteltu eteneminen, sekä tarvitseeko projekti lisää aikaa tai tulisiko se lopettaa. Vuonna 2012 VR:n uusi lipunmyyntijärjestelmä toteutettiin vesiputousmallilla ja

vasta projektin käyttöönotossa huomattiin järjestelmän olevan käyttökelvoton. Lipunmyyntijärjestelmä kaatui ensimmäisen tunnin jälkeen aamuruuhkassa. Kukaan ei voinut ostaa lippuja ja ongelmat tulivat esiin vasta tuotantokäytössä. Vesiputousmallin vuoksi virhettä ei havaittu aikaisemmin ja niitä ei pystytty korjaamaan ajoissa. Ohjelmisto oli alkuperäisestä päivämäärästä venynyt 2 kertaa puutteellisen ohjelmiston vuoksi. Perinteisessä työtavassa, jossa projektin sisältö ja aikataulutus on saneltu etukäteen ja kenties jopa vahvistettu tiukoin sopimuksin, törmätään ongelmiin. Tilaaja haluaa saada mahdollisimman paljon, toimittaja haluaa tehdä mahdollisimman vähän (Vanhala 2012). Kyseinen malli ei kovin usein sovi IT-projekteille sen jäykkyyden vuoksi.

3.2 Ketterä projektimalli

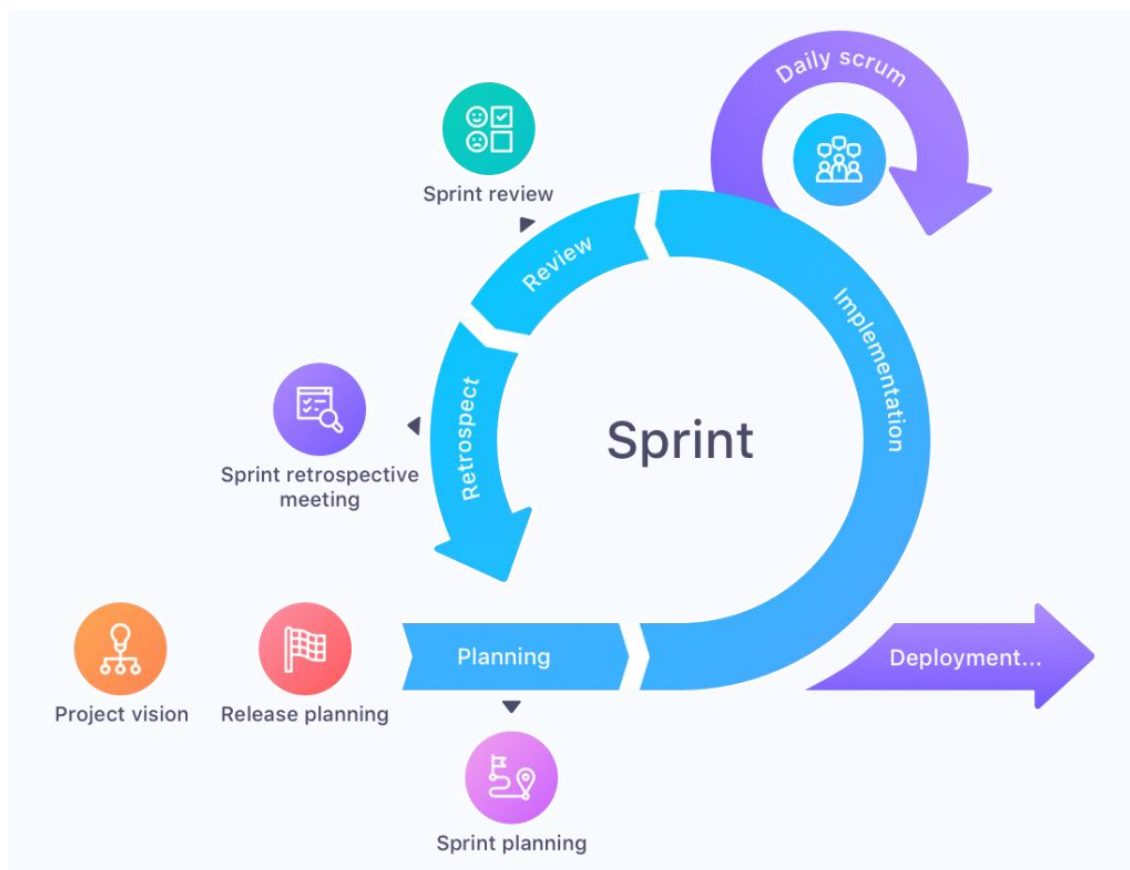
Ketterän kehitysmalli on saanut alkunsa Utahissa vuonna 2001, kun 17 mallin puolesta puhujaa kokoontuivat keskustelemaan kehitysmallista. Tilaisuutta pidetään ketterän ohjelmistokehityksen julistuksena, jossa korostettiin seuraavia asioita:

- Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
- Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
- Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

(Agile Alliance 2001)

Ketterät menetelmät pyrkivät ratkaisemaan ongelman, joka syntyi, kun rakennettavien ohjelmistojen vaatimukset ja halutut ominaisuudet muuttuivat nopeammin kuin projektit valmistuivat (Törnqvist 2015). Ketterän projektimallin tarkoituksena on pitää asiakas tyytyväisenä koko projektin ajan toimittamalla tarpeet täyttäviä versioita aikaisessa vaiheessa ja säännöllisesti. Yleensä suositetaan noin viikon tarkasteluväliä, jolloin asiakas pääsee kokeilemaan uutta versiota. Projektia on mahdollista muokata koko sen kehityksen ajan. Ketterän kehittämisen avulla dokumentaation merkitys pienenee ja tilaajan kanssa keskustelu kesken projektin kasvaa. Ketterässä kehityksessä projektin tehokkuutta mitataan projektin ajan ja tarpeen tullen sitä muokataan paremmaksi, esimerkiksi muuttamalla resursseja. Ketterät menetelmät sopivat parhaiten suuren epävarmuuden projekteihin, joiden lopputulos on uniikki. Paras tapa keskustella tiimin kesken projektissa on jokapäiväinen keskustelu ennen päiväntöiden aloittamista. Keskustelut projektin onnistumisesta käydään, vasta kuin projekti on tuotantokäytössä. Kuvassa 4 on ketterän

kehittämisen malli, jossa edetään sprinteissä ja tarvittaessa voidaan aina palata edelliseen osioon. (Engberg 2021) IT-projekteissa käytetään usein Scrum- tai Kanban-mallia.



Kuva 4. Ketterä projektimalli (Engberg 2021)

Scrum

Scrum-projektit pilkotaan pieniin Scrum-tiimeihin, johon kuuluu kehitystiimi, Scrum Master ja tuoteomistaja. Jokainen kehitettävä asia pilkotaan pieniin palasiin ja niille annetaan tiimin kesken aika-arviot. Scrum-tiimin kesken pilkotut tehtävät viedään tehtävälistalle ja priorisoidaan tärkeysjärjestykseen. Tämän jälkeen tiimit ottavat yhden jakson tehtävälistalle niin paljon töitä, kuin jaksolle mahtuu aika-arvioituja tehtäviä. Jaksojen pituus voi vaihdella, mutta usein käytetään 2 viikon jaksoja. Kun jakso tulee päätökseen, viedään valmiit tehtävät asiakkaalle ja aloitetaan sama kierros alusta. Scrum perustuu empirismiin ja lean-ajatteluun. Empirismin mukaan tieto tulee kokemuksesta ja päätösten tekemisestä havaintojen perusteella. Lean-ajattelu vähentää hukkaa ja keskittyy olennaiseen. (Schwaber ja Sutherland 2020)

Kanban

Kanban on hyvin samankaltainen tapa työskennellä kuin Scrum. Projektikokonaisuus jaetaan pieniin osiin ja toteutuneita tehtäviä tarkastellaan yleensä 2–3 viikon välein. Kanbanista puuttuu aikarajaukset tehtäviltä. Seuraava tehtävä valuu tehtävälialta automaattisesti edellisen tullessa valmiiksi. Usein projektin alku tehdään Scrum-mallin avulla, koska projektin tulee pysyä aikataulussa. Kanbaniin voidaan siirtyä projektin loppuvaiheessa, kun kyseessä on enää pieniä muutoksia ja ylläpitoa.

3.3 Projektin epäonnistumiseen vaikuttavat tekijät

Projektien epäonnistumiseen voi vaikuttaa useampi syy. Yleisimpänä ongelmana esiin tulee huonot ja epäselvät vaatimukset. Tilaaja ei tiedä mitä haluaa, ja toimittaja ei tiedä mitä pitäisi tehdä. Resurssipula sotkee projektien aikataulua, minkä vuoksi projekti ei valmistu ajoissa. Sama ongelma heijastuu projektin aikatauluihin alusta asti. Luvataan liian nopealla aikataululla, jossa ei huomioida mahdollisia ongelmia tai muutoksia. Oikeita resursseja ei löydetä projektiin tai resursseja on liian vähän. Asiakas ei ole tyytyväinen toteutettuun laatuun. Ohjelmien monimutkaisuus ja kompleksisuus tekee toteuttamisesta mahdottoman. Huono projektijohtaminen aiheuttaa usein epäonnistumia. Suunnittelun ja kommunikaation puute vesittää projektin ennen kuin se on ehtinyt alkaa. Tutkimusten mukaan suurin syy projektien epäonnistumiseen on puutteellinen viestintä. (Blomqvist 2019)

3.4 Suunnittelu

Projektinsuunnittelu koostuu useammasta vaiheesta, jotka tulee tehdä huolella. Jokainen organisaatio käyttää erimäärän aikaa ja resursseja suunnitteluvaiheessa, mutta seuraavia ongelmia tulisi välttää:

- Projekti ei vastaa olemassa olevia tarpeita ja asetettuja tavoitteita
- Projektitoimituksen kokonaisuutta ei saada toimimaan
- Projektisuunnitelma ei ohjaa tekemistä
- Projektiin ei sitouduta, kun sitä ei ole vaivauduttu edes suunnittelemaan.
- Projektin kohderyhmää (kenelle tehdään) ei osallisteta riittävän ajoissa
- Ei selkeytetä tarpeita
- Tehdään tarpeetonta lisätyötä tai ylilaatua
- Ei hallita projektiin liittyviä muutoksia
- Aliarvioidaan projektin monimutkaisuus.

(Mäntyneva, 2016, s 43)

Suunnittelussa tulee saada selville projektin tavoitteet ja rajata projekti hyvin. Liian väljät tavoitteet hankaloittavat suunnittelua ja arviointia. Projektin laajuus tulee lyödä lukkoon jo alkuvaiheessa, jotta tiedetään rajata projekti useampaan etappiin ja vaiheeseen.

Hyvä projektisuunnitelma kertoo, miksi ja miten projekti toteutetaan. Projektisuunnitelma toimii myös hyvänä dokumentaationa ja viestii tilaajalle onnistuneesta projektista. Projektisuunnitelman dokumentaatio on hyvä pitää selkeänä ja helposti ymmärrettävänä koko projektin ajan. Suunnitteluvaiheessa luodaan aikataulu vain kuukausi- tai vuosineljänneksitasolla. Karkean suunnitelman avulla on helpompi suunnitella projektin kokonaisnäkemyistä aikataulullisesti. Samalla pystytään luomaan suuntaa antavia työmääriä arvioita eri resursseille ja projektin vaiheille. (Mäntyneva 2016)

Projektisuunnitelman viimeistelyssä käytetään hyödyksi valmistusvaiheen selvityksiä ja hahmotelmia. Projektitoiminnan suunnitelmallisuutta voidaan edistää määrittelemällä suunnitelman laatimistapa, sisältö ja hyväksymiskäytännöt. Projektisuunnitelman tulee sisältää minimissään:

1. projektin tuotokset
2. projektin tehtävät
3. vastuunjako tehtävittäin
4. projektin aikataulutus
5. projektin budjetti.

(Mäntyneva 2016)

Projektisuunnitelmaa voidaan käyttää seurannan ja arvioinnin tukena. IT-projektin projektisuunnitelmassa tulisi ottaa huomioon seuraavat asiat.

1. Keskeiset liiketoiminnan käsitteet, jotta toimittajat ymmärtävät myös käsitteet.
2. Projektin tausta ja hyödyt
3. Projektin aikatauluttaminen
4. Budjetointi
5. Mittarit milloin projekti on onnistunut
6. Testaaminen ja hyväksymistestaaminen
7. Tilaaajan kouluttaminen
8. Käyttöönoton suunnittelu

IT-projekteissa tilaajille tulee usein yllätyksenä testaamiseen tarvittava aika ja miten sovellus saadaan tuotantokäyttöön stabiilisti toimivaksi. Tarkan projektisuunnitelman avulla pystytään välttämään kyseiset yllätykset. IT-projektin työvaiheiden kuormittavuudet jakautuvat usein seuraavasti:

- Määrittely ja suunnittelu 30 %
- Toteutus 40 %
- Testaus 30 %

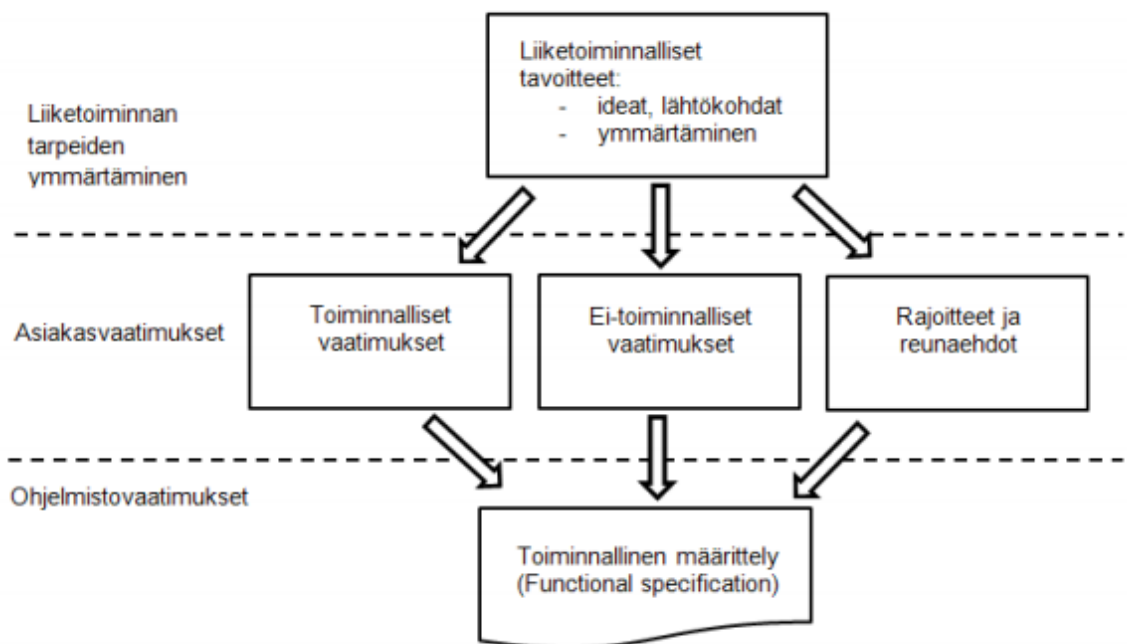
(Ruuska 2001)

Onnistuneessa projektisuunnitelmassa on saatu karsittua riskit minimiin, sekä toimittaja ja tilaajat tietävät, millainen tuleva tuote on valmiina. Hyvällä projektisuunnitelmalla voidaan säästää resursseja, aikaa sekä välttää ongelmia. Projektisuunnitelman tulee olla

myös joustava, mikäli ongelmia tulee sen edetessä. Projektisuunnitelma on projektin strateginen johtamistyökalu, siinä määritellään tavoitteet, organisointi, toimintamalli, tärkeimmät tuotokset ja työsuunnitelma sekä panokset sillä tarkkuudella, mitä rahoitus edellyttää.

3.5 Projektin vaatimusmäärittelyt ja elinkaari

Uusien IT-projektien suunnittelun jälkeen tehdään aina vaatimusmäärittelyt. Vaatimusmäärittelyissä käydään projektin vaiheet ja ominaisuudet läpi tarkemmin ja selvitetään tulevan tuotteen toiminallisuudet. Vaatimusmäärittelyt jaetaan usein kahteen eri kategoriaan, toiminnalliset ja ei-toiminnalliset vaatimukset. Vaatimusmäärittelyssä pyritään avaamaan ja dokumentoimaan jokainen toiminallisuus ja sen vaiheet. Käyttöliittymä puolella jokainen toiminto pyritään luomaan helppokäyttöiseksi ja tilaajaa ohjaavaksi. Kuvassa 5 kuvataan asiakas- ja ohjelmistovaatimukset. (Haikala & Mikkonen 2011)



Kuva 5 Asiakas- ja ohjelmistovaatimukset

Toiminnalliset vaatimukset määrittelevät tulevan kehitettävän ja hankittavan järjestelmän käyttämistä ja toiminnallisuutta. Ne määrittelevät ohjelmiston palvelut ja miten sen tulee toimia tietyissä tilanteissa. Kyseessä on joukko ominaisuuksia, jotka voidaan jaotella päätoimintojen alle. Toiminnallisissa määrityksissä luodaan koko ohjelmiston arkkitehtuuri ja asetetaan sen tekniset reunaehdot. Eli valitaan ohjelmistolle sopivat tekniikat ja miten niitä tulee käyttää, sekä millainen on tuotanto käytön tekninen ympäristö. Toiminnallinen määrittely kuvaa kaikki toiminnot, joita ohjelmiston tulisi tehdä ja arkkitehtuurisuunnittelu kuvaa ohjelmistokomponentit, jotka toteuttavat määrittelyn vaatimat toiminnot. (Haikala & Mikkonen)

Ei-toiminnalliset vaatimukset määrittelevät rajoitukset ja reunaehdot projektin toiminnallisille vaatimuksille. Toiminnot eivät suoraan liity toimintoihin, vaan kertovat ehdot toiminnallisille ominaisuuksille, miten niitä voidaan käyttää. Esimerkiksi ohjelmiston käyttöliitymä, asennukset, ohjeistukset, vasteaika, lokit ja arkistointi kuuluvat ei-toiminnallisiin ominaisuuksiin. Ei-toiminnallisilla vaatimuksilla on kriittinen rooli tietojärjestelmän laadun, luotettavuuden ja kustannustehokkuuden takaamisessa koko elinkaaren ajan. (Lepänen 2020)

Projektin elinkaaren määrittäminen on hyvä tapa selvittää projektin kokonaiskuva. Projektilla pitää aina olla alkamis- ja päättymisajankohta, jotka muodostavat kokonaiskeston. Projektin elinkaari jaetaan usein neljään päävaiheeseen:

1. valmistelu
2. suunnittelu
3. toteuttaminen
4. päättäminen.

Valmisteluvaiheessa rajataan projektin laajuus ja selvitetään kohderyhmät. Suunnitteluvaiheessa pyritään saamaan projektille rahoittaja. Tilatuissa projekteissa sopimukset kirjoitetaan usein suunnitteluvaiheessa, joten tilaajan ja toimittajan tulee ymmärtää mitä projektilla saavutetaan. Sisäisissä kehitys- ja tuotekehitysprojekteissa organisaatiot itse päättävät projektin toimeenpanosta ja budjetista. Huolella hoidettu valmisteluvaihe helpottaa etenemistä varsinaiseen projektin suunnitteluun. (Mäntyneva 2016)

Suunnitteluvaiheessa projektin määrittelyt tehdään tarpeeksi yksityiskohtaiseksi, jotta pystytään selvittämään eri vaihtoehtoja projektin toteuttamista varten. Projektille pyritään

löytämään helpoin ja edullisin tapa toteuttaa se määritelmien mukaisesti. Tehtävät jaetaan resursseille ja pystytään muodostamaan tarkempi aikataulu koko projektille. Suunnitteluvaiheessa on tärkeä kyetä suunnittelemaan projektin aikataulu, kustannukset ja resurssit riittävän tarkasti. (Mäntyneva 2016) Suunnitteluvaiheessa kaikki dokumentoidaan erilliseen projektisuunnitelmaan ja riskit sekä ongelmakohdat tulisi tunnistaa jo suunnitteluvaiheessa.

3.6 Toteutus

Projektin toteutusvaiheessa toteutetaan projektisuunnitelmassa kuvattu projekti. Mikäli projektin aikana ilmenee ongelmia, pitää projektisuunnitelman olla valmis muutettavaksi. Jokainen muutos ja ongelma tulee dokumentoida. Projektia seurataan ja pyritään selvittämään ongelmakohdat mahdollisimman nopeasti esimerkiksi lisäämällä tai muuttamalla resursseja. Projektin aikana tulee aina muutoksia projektisuunnitelmaan. Yleisimpiä poikkeamia ovat resurssimuutokset, tilaajan muutokset, aikataulumuutokset ja tehtävän keston väärin arviointi. Suurin osa projekteista ei pysy aikataulussa. Tämä johtuu usein heikosta seurannasta ja liian tiukoista aikatauluista. Jos suunnittelu- ja testausvaiheessa huomataan, että jokin asia ei toimi toivotulla tavalla, on pystyttävä kehittämään vaihtoehtoisia toimintatapoja. (Mitäh 2019)

Viestinnän tärkeys tulee esiin projektin toteutusvaiheessa. Jokaisesta myöhästymisestä tai muutoksesta tulee heti kertoa projektipäällikölle. Useat tehtävät ja resurssit voivat olla riippuvaisia tietyistä ominaisuuksista. Hyvä viestintä antaa mahdollisuuden uudelleen järjestää aikatauluja ja tehtäviä.

Projektin talouden tulee myös pysyä kunnossa, jotta päästään projektin tavoitteisiin. Projektipäällikön tulee seurata kuluja ja projektin budjettia jatkuvasti. Mikäli rahat loppuvat liian aikaisin, voi projekti epäonnistua ja jäädä valmistumatta. Projektin kustannusten kertymistä on tarpeen verrata projektin etenemiseen ja tuloksiin. (Mäntyneva 2016) Projektin aikana on usein myös tulovirtaa, kun tilaajaa voidaan laskuttaa projektin eri vaiheissa.

Projektien edetessä tilaajalle tulee usein eteen muutosten tarve tai muutoksia voidaan joutua tekemään lainsäädännöllisistä syistä. IT-projekteissa muutoksilta ei voida välttyä. Ketterä projektimalli auttaa tilaajaa löytämään muutostarpeet usein kesken projektin ja niiden muuttaminen on usein helpompaa kuin lopputestauksessa. Projektiin liittyvien muutostarpeiden osalta voidaan toimia neljällä tavalla. Projektin aikana tunnistetut muutokset tulee huomioida välittömästi ja reagoida niihin muutoksen vaatimalla prioriteetilla. Muutokset voidaan hyväksyä tietyin reunaehdoin, eli mitkä on mahdollista vielä toteuttaa kyseisessä projektissa. Muutokset voidaan tarpeen tulleen jättää myöhemmäksi ja ne lisätään projektiin viiveillä. Viimeisenä mahdollisuutena on hylätä kaikki tunnistetut muutospyyntö.

3.7 Testaus

Testaus vaiheessa projekti käydään läpi usein ohjelmallisilla testiohjelmilla ja erillisen testausryhmän kanssa. Kun jokainen virhe on saatu korjattua, voidaan projekti siirtää tuotantotestaukseen. Usein tilaaja on saanut testata ohjelmaa jo sen kehitysvaiheessa, mikäli projektissa on käytetty ketterää projektimallia. Tuotantotestauksessa usein havaitaan käytännön ongelmia, kun oikeat loppukäyttäjät saavat testata ohjelmaa. IT-projekteissa tulee tehdä testaus suunnitelmat, jotta tiedetään mitä testataan. Osa IT-projekteista on suunniteltu testaus edellä. Ensimmäinen on kirjoitettu automaatiotestit ohjelmalle ja sen jälkeen ohjelmoitu lopputuote. Kyseisellä tavalla pyritään luomaan mahdollisimman toimiva kokonaisuus kerralla. Kokemus osoittaa, että toimivien ja helppokäyttöisten järjestelmien tekeminen edellyttää jatkuvaa testaamista ja koekäyttäjien osallistumista. (Vanhalala 2012) Huono testaus ja ongelmien heikot ratkaisut aiheuttavat projektin epäonnistumisen.

3.8 Projektin päättäminen

Projektin viimeisenä vaiheena on projektin päättäminen. Projekti on saatu valmiiksi määritelmien mukaisesti ja projektipäällikkö on laatinut loppuraportin, jossa dokumentoidaan projektin tulokset ja kerrotaan, miten projekti lopulta onnistui. Loppuraportti on tiivis dokumentaatio projektista ja kehityksen edetessä syntyneistä ongelmista ja niiden ratkaisuista. Lopuksi projektinohjausryhmä käy läpi koko projektin tavoitteet ja sen, miten tavoitteet ovat onnistuneet. Projekti tulee lopettaa oikein. Projektipäällikkö vie IT-projektin

loppuun asti täydellä panostuksella, jottei koko tiimin ote herpaannu projektista. Muuten resursseja saattaa jäädä roikkumaan projektiin ja se vie turhaa työaika muilta projekteilta. IT-projekteissa tulee tehdä epäonnistuneestakin projektista päätösraportti, muuten epäonnistumisen syyt eivät koskaan selviä. (Lind 2001)

3.9 Projektin jatkokehitys

IT-projekteissa tulee aina eritellä projektin loppu- ja jatkokehitystehtävät; muuten projekti ei pääty koskaan. Projektin päättyttyä projektipäällikkö laatii asiasta lopputiedotteen, josta käy lyhyesti ilmi, mitä projektissa tehtiin, tarvittavat jatkotoimenpiteet sekä ketkä vastaavat lopputuotteen teknisestä tuesta, ylläpidosta ja jatkokehityksestä. Jatkokehitystehtävät alkavat usein hyvin nopeasti projektin valmistuttua, kun käyttäjät ovat saaneet antaa palautetta lopputuloksesta. Jatkokehitystehtävien laajuus voi vaihdella pienistä ulkoasumuutoksista suuriin projekteihin, jotka liittyvät alkuperäiseen projektiin.

4 RYHMÄHAASTATTELU

Opinnäytetyössä käytettiin aineistonkeruumenetelmänä ryhmähaastattelua. Haastattelun avulla pyrittiin saamaan mahdollisimman paljon aineistoa tulevaa projektia varten. Vastaukset kerättiin fläppitaululle, mistä vastauksia käytiin yhdessä läpi. Tämän jälkeen niistä valittiin tärkeimpiä ominaisuuksia projektisuunnitelmaan. Työhön valittiin ryhmähaastattelu yksilöhaastattelujen sijaan, koska keskustelun oletettiin tuovan enemmän näkökulmia ohjelman kehittämisestä ja sen ominaisuuksista. Lisäksi tavoitteena oli arvioida aikaisempaa versiota ja sen toiminnallisuuksia. Ryhmähaastattelun tehtävänä on mahdollistaa uusien innovaatioiden luominen ja hyödyntäminen uudessa ohjelmassa. Haastattelut ovat sopiva tapa, kun on olemassa tarve kerätä perusteellista tietoa ihmisten mielipiteistä ajatuksia, kokemuksia ja tunteita. (Virginia Tech 2018).

Tutkimuksen avulla haluttiin luoda asiakkaille tunne, jossa he saavat itse vaikuttaa asioihin alusta asti ja heitä kuunnellaan. Kommunikoinnin oletettiin olevan helpompaa projektin aikana ja muiden tarvitsemat muutokset voitiin ottaa helpommin vastaan. Tämänhetkessä ohjelmassa on paljon asiakaskohtaisia ominaisuuksia, joista muut asiakkaat eivät välttämättä tiedä. Kyseisistä ominaisuuksista pyritään pääsemään eroon tai yhdistämään ominaisuuksia. Ryhmässä ideoita syntyy paremmin, ja kasvokkain kommunikointi on tehokas tapa kehittää asioita. Ryhmäkeskustelujen erityinen anti liittyy osallistujien väliseen vuorovaikutukseen ja siinä toteuttavaan yhteisen tiedon tuottamisen prosessiin. (Hyvärinen & Ruusuvuori 2017)

4.1 Tutkimusongelma

Tämän kehittämistyön tavoitteena ja tutkimusongelmana on suunnitella ja toteuttaa uusi tapahtumahallinnan ohjelma, joka on nykyaikaisempi, käytettävämpi ja ylläpidettävämpi ohjelma asiakkaille. Mikä on ohjelmiston nykytilanne, nykyiset ongelmat ja miten ne voitaisiin ratkaista? Ryhmähaastatteluiden kysymyksillä pyrittiin selvittämään nykytilannetta, ja edelleen ryhmätyöskentelyn avulla saamaan selville nykyiset ongelmat, ja löytämään ratkaisut, miten ne voidaan jatkossa toteuttaa paremmin. Yritykselle pyritään luomaan uusia tapoja kehittää projekteja asiakkaiden kanssa.

4.2 Haastattelututkimus

Opinnäytetyön tutkimustapana käytetään ryhmähaastatteluja, joiden tulosten perusteella pyritään luomaan projektin määrytykset ja tarpeet. Samalla pystytään analysoimaan projektin tarpeita ja haluttuja muutoksia edellisestä versiosta. Tavoitteena on pystyä luomaan kyseisten tulosten perusteella jokaiselle käyttäjälle sopiva ohjelma ilman asiakas-kohtaisia ominaisuuksia.

Kyseiseen haastattelutilaisuuteen kutsuttiin 15 eri yritysten pääkäyttäjää ja hallintakäyttäjää, joiden kanssa on myös kehitetty järjestelmää tälläkin hetkellä, mutta tutkimuksen avulla oli tarkoitus tehostaa ohjelmistoa ja käytettävyyttä huomattavasti. Haastatteluun saapui 75 %:a kutsutuista pääkäyttäjistä. Haastattelun pelkona oli pieni osallistujamäärä, joka johtaisi ryhmien vähenemiseen. Poisjäännit johtuivat työkiireistä tai pääkäyttäjällä ei ollut vielä tarpeeksi kokemusta sovelluksesta. Mukana on julkisen ja yksityisen sektorin yritysten edustajia eri kokoisista organisaatioista. Haastattelut toteutettiin 4 hengen ryhmissä ja jokaisessa ryhmässä oli edustettuna yrityksen asiantuntija, jonka tehtävänä oli tehdä havaintoja ja tuoda omia näkemyksiään tarvittaessa. Kysymykset olivat avoimia kysymyksiä, joihin ryhmät saivat vastata haluamallaan tavalla. Tutkimukseen osallistuvat yritykset olivat kooltaan 100 työntekijän ja 40 000 työntekijän väliltä.

Tutkimus ei sellaisenaan kerro ohjelmiston nykytilaa tai puutteita. Myöskään uutta ohjelmistoa ei voida suoraan toteuttaa tulosten perusteella, mutta tarkoituksena on saada avainominaisuudet projektille. Ryhmähaastattelussa on myös omat haasteensa, miten saadaan jokaiselta tarvittavat tiedot. Ryhmässä saattaa olla usein yksi voimakas persoona, joka pystyy ajamaan yksin koko ryhmän asioita. Mikäli joku henkilö olisi vienyt ryhmää liikaa yksinään eteenpäin, oli yrityksen oman työntekijän autettava ja ohjattava muita mukaan keskusteluun. Haastattelun tarkoitus oli saada mahdollisimman monta eri näkökulmaa ja työskentelytapaa, jotta tutkimusta voidaan hyödyntää suunnittelu- ja määrittelyvaiheessa.

Ryhmät vastasivat ensin kysymyksiin, jonka jälkeen yhdistettiin ryhmien tuloksia ja esiin tulleita tärkeimpiä ominaisuuksia yhdessä kaikkien ryhmien kesken. Jokaisen ryhmän tuli valita kolme tärkeintä ominaisuutta, jotka tulisi huomioida uudessa projektissa. Tämän jälkeen yhdessä keskusteltiin valituista ominaisuuksista ja pohdittiin, voidaanko

ominaisuuksia yhdistää tai toteuttaa toisen ryhmän ehdottamalla tavalla. Haastateltavien työskentelytyylit poikkesivat tiettyjen osa-alueiden osalta, mikä näkyi niiden osioiden keskusteluissa. Kyseisiä ominaisuuksia olisi ollut hankala yhdistellä ilman yrityksen asiantuntija apua. Hän pystyi näkemään asian molempien osapuolien näkökulmasta ja osasi kertoa, miten sen voisi tulevaisuudessa toteuttaa paremmin ja tehokkaammin.

4.3 Haastattelun toteutus

Tutkimuskysymyksillä pyrittiin selvittämään uuden projektin määritykset ja tarpeet. Kysymykset ja tehtävät oli pohdittu yhdessä projektipäälliköiden ja ryhmien vetäjien kanssa. Kysymykset ja ryhmätyöt olivat jaettu seuraaviin vaiheisiin:

- Nykyisen käyttötarkoitus/ ongelmat
- Mitä uusia käyttötilanteita ohjelma voisi tarjota
- Vastausten ryhmittely
- Uusien ominaisuuksien pohtiminen
- Toisten ryhmien palautteiden kirjaus ja niiden läpikäynti

Kysymykset haluttiin pitää mahdollisimman avoimina ja niiden toivottiin herättävän keskustelua ryhmissä. Kysymyksillä haluttiin luoda ensin keskustelua nykyisistä tarpeista ja ohjelman puutteista. Keskustelun alettua ryhmien on helpompi alkaa pohtimaan kehitystarpeita uudelle sovellukselle. Kysymysten pohtimiseen ryhmille annettiin aikaa yleensä noin 10–30 min riippuen kysymyksestä. Haastattelu haluttiin pitää intensiivisenä keskusteluna, jotta mielenkiinto pysyy yllä. Kyseinen aika oli jossain kysymyksissä hieman lyhyt, koska ohjelma on niin laaja ja jokaisella yrityksellä on hieman eri ominaisuuksia käytössään. Ensimmäiset minuutit menivät helposti ominaisuuksien vertailuun, eikä niiden tulevaisuuden suunnitteluun.

Tutkimuksen kysymyksillä pyrittiin aluksi selvittämään jokaiselta ryhmältä, miten he käyttävät nykyistä järjestelmää. Nykyistä järjestelmää tarjotaan asiakkaille räätälöityinä kokonaisuuksina, minkä vuoksi käyttötarkoituksia voi olla useampia. Kysymyksellä pyrittiin samaan aikaan keskustelua ryhmissä ja pohtimaan nykyistä järjestelmää, sekä tukemaan tulevia kysymyksiä. Vastausten perusteella ryhmät olivat selkeästi tyytyväisiä nykyiseen ohjelmaan, mutta pitivät sitä hieman vanhanaikaisena. Ohjelman yleisilme vaatisi muutoksia, varsinkin nuoremman sukupolven käyttäjien mielestä. Painikkeiden ja tekstien tueksi toivottiin kuvia ja logoja, jotka ohjasivat käyttäjää eteenpäin.

Seuraavaksi ryhmiä pyydettiin pohtimaan, millaisissa uusissa käyttötilanteissa uusi ohjelma voisi palvella nykyistä koulutusten hallintaohjelmistoa. Tämän jälkeen vastaukset tuli ryhmitellä käyttötapaukset ja ryhmitellä ne otsikoiden alle.

Viimeisenä tutkimuksessa ryhmiä pyydettiin pohtimaan uusia ominaisuuksia uuteen sovellukseen ja perustelemaan niiden hyötyjä. Ideat laitettiin otsikoiden alle fläppitaululle ja muut ryhmät tulivat antamaan palautetta toiselle ryhmälle. Ryhmien palauteet purettiin yhdessä vielä fläppitaululle, johon myös yrityksen henkilöstö antoi oman näkemyksensä muutoksiin ja uusiin ominaisuuksiin. Kyseinen data kerättiin samalla suoraan projektisuunnitelmaan.

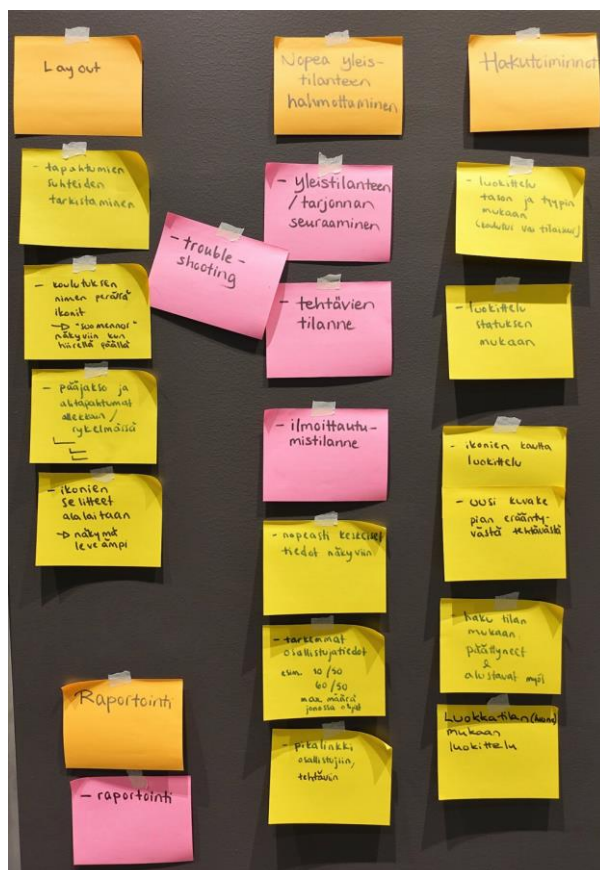
4.4 Haastattelun tulokset

Haastattelussa mukanaolleet pääkäyttäjät olivat sekä kokeneita tapahtumahallinnan käyttäjiä tai vasta-aloittaneita käyttäjiä. Tämä näkyi ryhmätyön haastattelussa, tuloksena saatiin useita uusia näkökulmia, joita aloitettiin läpi käymään projektipäälliköiden ja toimitusjohtajan kanssa. Läpikäynnissä todettiin ryhmähaastattelun olleen positiivinen kokemus molemmille osapuolille. Haastattelusta saatiin kaikki tarpeellinen aineisto uutta ohjelmaa varten. Ryhmissä ei jääty väittelemään ominaisuuksien tarpeellisuudesta, vaan pyrittiin miettimään, miten kyseisen ominaisuutta voisi itse tulvaisuudessa hyödyntää. Tarvittavien ominaisuuksien määrä oli valtava, joka tulee hankaloittamaan määrittelyiden priorisointia projektissa.

Tulosten perusteella aloitettiin tekemään projektin määrittely ja päättämään sen aikataulusta. Muutoksia ja uusia ominaisuuksia tuli todella paljon, joten ensimmäisenä piti poimia tärkeimpiä asioita, jotka tuodaan ensimmäiseen versioon. Moni tutkimuksessa tärkeänä pidetty ominaisuus perusteltiin, jotta jokapäiväiset työt helpottuisivat ja rutiinityöt nopeutuisivat. Tutkimus kuitenkin osoittaa, että lähes kaikki tarpeellinen löytyy jo nykyisestä ohjelmasta, mutta ominaisuuksien käyttöä tulisi helpottaa. Kuvassa 7 on haastattelussa esiin tulleita kommentteja ja ominaisuuksia uuteen sovellukseen. Kyseiset ominaisuudet olivat asiakkaiden ja yrityksen mielestä tärkeimpiä:

- Raportointi
- Tiedonhaku/seuranta ilmoittautuneista
- Kokonaistarjonnan tarkastelu
- Koulutustilaisuuden luominen

- Hakemusten käsittely
- Mahdolliset jonopaikat
- Tiedostojen liittäminen
- Koulutuksen päättäminen
- Tietoja käyttäjäaktiivisuudesta
- Tarve peruuttaa kurssi
- Osallistujien siirto kurssilta toiselle
- Tarve markkinoida/kasvattaa täyttöastetta
- Haku kohderyhmällä



Kuva 6. Haastattelun ominaisuudet jaoteltu otsikoiden alle

Asiakkaiden palaute kyseisestä haastattelutapahtumasta oli positiivista. Heille oli tärkeää päästä vaikuttamaan asioihin heti alusta ja vuorovaikutus avasi myös asiakkaan näkökulmia kehittämiseksi. Toivottiin samanlaista toimintamallia myös tulevaisuudessa uusien projektien kanssa. Moni aikoo jatkossa käyttää samantyylistä ryhmähaastattelua

myös oman yrityksensä kehittämisprojekteissa. Uudelta projektilta oletetaan myös parannusta ohjelman laadussa ja toimivuudessa. Kyseiseen ongelmaan pyrittiin saamaan vastaus kyseisestä haastattelusta. Tutkimuksessa tuli esiin paljon uusia ominaisuuksien tarpeita ja vanhojen kehittämiseen liittyviä muutoksia. Ongelmana tulee olemaan se, miten saadaan kaikki asiat toteutettua uuteen ohjelmaan järkevässä ja ajassa ja poimittua sieltä tärkeimmät ominaisuudet. Kyseisen tutkimuksen jälkeen monelle saattaa jäädä mielikuva uudesta ohjelmasta ja osallistujat saattavat helposti olettaa siitä löytyvän kaikki haluansa ominaisuudet, mitä he ovat projektin aikana tuoneet esiin.

5 VAATIMUSMÄÄRITTELY JA TOTEUTUS

5.1 Nykyinen koulutuksenhallinohjelmisto

Yritys tarjoaa koulutusten ja osaamisen hallintaohjelmistoa sitä tarvitseville asiakkaille. Tässä opinnäytetyössä kehitetään koulutuksen hallintaohjelmisto-osuutta. Koulutus ja osaamisen kehittäminen ovat organisaation kilpailukyvyyn ylläpitämistä ja aineettoman pääoman rakentamista. Koulutuksenhallinnan avulla asiakkaille tarjotaan mahdollisimman nopeaa ja suoraviivaista koulutusten hallintaa.

Järjestelmä on täysin selainpohjainen ja modulaarinen palvelu, jota voidaan hyödyntää myös osaamisenhallinnan puolella. Järjestelmää pystyy käyttämään kaikilla nykypäivän yleisimmillä internet-selaimilla. Järjestelmä on toteutettu monisivuisena web-projektina. Se on toteutettu Microsoft .NET Frameworkin kanssa.

Järjestelmässä pystytään luomaan ja kopiomaan kursseja, tilaisuuksia tai verkkokursseja. Samalla hallitaan kurssien osallistujia ja heidän osaamisensa kehittymistä. Koulutuksen hallinnan avulla pystytään tuottamaan valmiita vuosiraportteja tapahtumista ja niiden osallistujamääristä. Myös viestintä ja koulutusmateriaalin sekä muun aineiston jakaminen osallistujille onnistuu palvelun avulla. Osallistujat voivat seurata koulutuksen ohjelmaa reaaliaikaisesti tapahtuman aikana, ja järjestelmä antaa heidän luoda tapaamisia ja verkostoitua keskenään. Ratkaisu tekee koulutuksesta itseohjautuvaa, mikä vähentää hallinnollista työtä ja sitouttaa osallistujia. Henkilöstö ja muut osallistujat näkevät oman koulutushistoriansa ja tulevat koulutuksensa Omapalvelu-portaalin kautta. Järjestelmä tukee myös laskutusta.

5.2 Kehittämiskohteiden valinta

Kehittämiskohteita pohdittiin yrityksessä sisäisesti 4 hengen ryhmässä. Aluksi päätettiin millä frameworkilla kyseinen projekti toteutetaan. Tässä päädyttiin Angular 2 ja .Net Core ratkaisuun. Kyseinen ohjelma haluttiin toteuttaa uusimmilla tekniikoilla, vaikka tämä vaatii tekijöiltä uusien asioiden opettelua. Angular frameworkin avulla ohjelmasta saadaan nopeampi ja käyttäjäystävällisempi toteutus. Samalla saadaan tuotua uutta osaamista yritykseen.

Tärkeimpänä ominaisuutena pidettiin asiakkaiden ja yrityksen projektiryhmän mielestä käyttöliittymää, jossa näkyy asioita mahdollisimman paljon ja selkeästi. Käyttöliittymästä päätettiin kojelautatyypinen ratkaisu, joka koostuu tietokortista. Vanhassa käyttöliittymässä käytettiin listoja, joista avattiin uusia sivuja. Kojelautalautamallin avulla tietoa saadaan näytölle näkyviin mahdollisimman paljon kerralla. Tapahtumien hallinta, muokkaaminen ja seuraaminen helpottuvat pääkäyttäjille, koska tärkeimmät asiat ovat jatkuvasti esillä. Tapahtumien hakemista ja lajittelua halutaan myös parantaa kojelautanäkymässä. Tapahtumien listauksessa päädyttiin käyttämään Telerikin Kendo-kirjastosta taulukko-komponenttia. Komponentti itsestään sisältää hyvät haku- ja suodatusmahdollisuudet ja sitä on myös mahdollista muokata tarvittaessa. Käyttöliittymä koostuu yhdestä päätaulukosta, josta valitaan haluttu tapahtuma. Valinta avaa kortit kojelautanäkymään ja sisältää tapahtumien tietoja.

Ensimmäiseen versioon haluttiin valita ominaisuuksia, joita hallintakäyttäjät tarvitsevat jokapäiväisessä työssä. Yhteen korttiin tuodaan tapahtuman perustiedot ja niiden muokausmahdollisuus. Toiseen korttiin valittiin osallistujien hallinta, jossa voidaan muokata osallistuja ja niiden tiloja. Hallintakäyttäjät joutuvat päivittäin vaihtelevaan henkilöiden tiloja ja ilmoittautumisen asetuksia. Palauteet ja tiedosto päätettiin ottaa mukaan vielä ensimmäiseen versioon omina kortteina. Kumpikin kortti toimii pääasiassa vain datan näyttämiseen. Nykyisen sovelluksen laajuuden vuoksi, emme pysty korvaamaan koko sovellusta uudella heti, mutta uuden ohjelman kautta siirtymistä vanhalle puolelle helpotetaan. Käyttäjä pystytään ohjaamaan oikeaan paikkaan yhden klikkauksen jälkeen, kun vanhassa sovelluksessa vaadittiin usein monia klikkauksia.

Tekstiviestien ja sähköpostien lähettäminen on tärkeää pääkäyttäjien jokapäiväisessä työssä. Molemmat ominaisuudet tuodaan uudessa ohjelmassa esiin modal-ikkunoiden kautta, joissa ohjelman päälle avataan uusi ikkuna. Modalin avulla saadaan taas koko ruutu käyttöön ja taustalta ei häviä käyttäjän aikaisemmat työt.

Projektia viedään eteenpäin kuvan 7 mallin mukaisesti. Jokaisessa kohdassa mahdollistetaan viestitä ja kuunnellaan mahdolliset muutospyynnöt asiakkailta. Projekti halutaan pitää joustava käyttöönottoon asti.



Kuva 7. Projektin etenemismalli

5.3 Käyttöliittymäsuunnittelu

Tämänhetkisessä käyttöliittymässä ongelmana pidettiin hitautta ja sitä, että oleellista tietoa löysi usein vasta useamman klikkauksen jälkeen. Käyttöliittymän ulkoasua pidettiin myös hieman vanhana ja sitä haluttiin samalla uudistaa. Vanhassa käyttöliittymässä tiedot tuodaan esiin erilaisista listoista ja niiden avulla siirrytään seuraavaan lista näkymään. Kuvassa 8 on kuva vanhasta käyttöliittymästä, jossa kurssit ovat listattu.

Yksikkö

Ei valittu

Näytä: Tapahtumatyyppi: Kenttä: Arvo:

Kaikki Koulutus Nimi Hakusanat

Alkaa aikavälillä:

22.10.2019 21.10.2021

Tulevat tapahtumat **Tänään on 21.10.2020**

23.10-22.11.2020 Yksikkökohtainen perehdytys, talous,	+u
30.10-30.10.2020 Valmentava esimiestyö, Aviabulevardi	+u
01.11-02.11.2020 1. jakso Asiakkuuksien ja myynnin johtamisen dynamiikka,	+u
02.11-02.12.2020 Yksikkökohtainen perehdytys, hallinto,	+u

Kuva 8. Vanhan käyttöliittymän listanäkymä

5.4 Rajapinnan suunnittelu

Ohjelmalle suunnitellaan myös rajapinta, joka kommunikoi käyttöliittymässä. Rajapintaa kutsutaan API:ksi. Tässä projektissa API rajapinnan avulla tuodaan data käyttöliittymäpuolelle. Kyseessä on ohjelma, jossa käsitellään suuria määriä dataa. Tietokantakyselyt päätettiin toteuttaa Entity Frameworkin avulla. Entity Frameworkin avulla pystytään parantamaan myös sovelluksen tietoturvaa. Tietokantaan tehdyt hyökkäykset hankaloituvat.

Tietokantakyselyjen nopeuteen ja käytettävyyteen halutaan panostaa uudessa ohjelmassa, koska hitaudet näkyvät heti loppukäyttäjälle. API rajapinnan avulla sovelluksen laajentaminen on tulevaisuudessa helpompaa.

5.5 Toteutus ja testaus

Projekti tehtiin ketterää kehitysmallia soveltaen. Varsinaisesti sprintit olivat välillä pitkiä, koska projektin ohella resursseja jouduttiin ajoittain käyttämään muualla. Uuden ohjelmisto kielen, arkkitehtuurin ja toiminnallisuuksien opiskelu viivästytti projektin aloitusta.

Projektin alun jälkeen, se eteni loogisesti ja suunnitelmien mukaisesti. Projektin edetessä valittiin alkuun 2 asiakasta, jotka saivat alusta asti kommentoida ja testata ohjelmaa. Asiakkaiden ja projektiryhmän välillä oli hyvää kommunikointia ja viestintää koko projektin ja sen suunnitteluajan ajan. Ohjelman kehitys- ja toteutusvaiheessa ei varsinaisesti muutettu kuin nappien tekstejä ja logoja.

Projektiryhmän jäsenet ja asiakkaiden käyttäjät testasivat ohjelmaa koko ajan, kun uusia ominaisuuksia tuli. Virheet pyrittiin korjaamaan aina ennen seuraava ominaisuutta. Projektin läpäistyä asiakkaiden viimeiset testaukset, ohjelma vietiin tuotantoon. Tässä vaiheessa ohjelma asennettiin halukkaille asiakkaille myös tuotantoon rinnakkaiskäyttöön vanhan sovelluksen kanssa. Käyttäjämäärien ja datan kasvaessa, alkoi esiintyä erilaisia ongelmia nopeudessa, joita ei voitu havaita ennen oikeaa käyttöä. Korjaaminen aloitettiin välittömästi ja kaikkia SQL-hakuja optimoitiin nopeammaksi ja poistettiin ylimääräistä dataa. Asiakastestauksen heikkoutena tuli esiin asiakkaan ahkeruus testata uutta ohjelmaa. Usein kokeillaan muutamaan ominaisuutta ja sanotaan toimivan. Hitausongelmat

tulivat ilmi vasta, kun useampi asiakas alkoi käyttää ohjelmaa. Projektiryhmän jäsenille testaamista hankaloittaa oletamus toimivuudesta ja liian hyvä tuntemus sovelluksesta; ei osata etsiä virheitä oikealla tavalla, esimerkiksi syöttää vääriä arvoja tiettyihin kenttiin.

Rinnakkaiskäytön avulla korjaukset oli helppo toteuttaa, koska se ei kuitenkaan estänyt asiakkaalta vanhan sovelluksen käyttöä. Optimoinnin jälkeen ohjelmakokonaisuudesta tuli todella toimiva asiakkaiden ja projektiryhmän mielestä.

5.6 Palaute ja kehitysehdotukset

Uudesta ohjelmasta on kerätty käyttäjäpalautetta sen valmistumisen jälkeen. Pääsääntöisesti uudistuksiin oltiin todella tyytyväisiä. Moni kuitenkin toivoo tiettyjen ominaisuuksien lisäämistä jatkokehitysvaiheessa, jotta he voivat kokonaan siirtyä käyttämään uutta ohjelmaa. Asiakkaat olivat tyytyväisiä, kun saivat olla projektissa mukana alusta asti ja vaikuttaa asioihin. Samalla toivottiin uudistusta myös ohjelman muihin osiin ja mielelletään samalla kehitysmenetelmällä. Asiakkaat pitivät haasattelututkimusta määrittelyjen tekemisessä niin onnistuneena, että osa aikoo käyttää samaa tekniikkaa tulevissa projekteissaan.

Ohjelman suuren datan näyttäminen ja suodattaminen on ollut asiakkaiden mielestä erinomainen uudistus. Kurssit ja tapahtumat voidaan hakea monella eri hakuriteerillä ja samalla pystytään vertaamaan tietoja toisen kurssin kanssa. Käyttäjäkohtainen näkymä on helpottanut monen työtä, koska työntekijät tarvitsevat eri asoita ruudulla kerrallaan. Jokaisella on kuitenkin mahdollisuus saada sama näkymä kuin muilla. Näytöllä ei myöskään tarvitse pitää mitään ylimääräistä esillä.

Kehitysehdotukset painottuvat enimmäkseen raportointipuolelle, joka jäi ensimmäisestä versiosta pois. Uudessa ohjelmassa saadaan tulostettua osallistujalistoja ja tapahtumalistoja, mutta yksityiskohtaisemmat raportit puuttuvat vielä. Kurssien luomiseen ja kopiointiin toivottiin uudistusta seuraavassa versiossa. Kurssien markkinointiin halutaan jatkossa mahdollisesti oma työkalu tai osio.

6 POHDINTA JA JOHTOPÄÄTÖKSET

Tässä opinnäytetyössä suunniteltiin ja toteutettiin uusi tapahtumahallinnanohjelma yritykselle, sekä kehitettiin projektityöskentelyä yrityksessä. Kehittämistyössä hyödynnettiin asiakkaita ja heidän tarpeitaan, jotta ohjelma palvelisi asiakkaita paremmin heidän työssään. Tässä luvussa käydään läpi, mitkä asiat onnistuvat hyvin ja mistä tuli lisätyötä.

Tutkimukseen kutsuttiin 18 asiakasta eri yrityksistä, joista 15 yrityksen pääkäyttäjä saapui paikalle. Koska osanottoprosentti oli niin suuri, oletettiin asiakkaiden olevan kiinnostuneita kehittämään sovellusta yhdessä. Haastattelu tuotti paljon erilaisia vastauksia ja paljon uusia ominaisuuksia uuteen ohjelmaan. Tutkimuksen herättämä mielenkiinto asiakkaissa oli positiivinen yllätys, joka johtuu hyvästä työskentelystä asiakasrajapinnassa. Asiakkaiden halukkuus sitoutua kehittämään uutta projektia, antaa positiivisen kuvan yrityksen asiakastytyväisyydestä ja siitä, että asiakkaat luottavat yritykseen jatkossakin. Tutkimuksesta saadut kehitystoiveista ja muutoksista jäi kuitenkin lopulta yrityksen projektitiimille päätettäväksi, mitä otetaan mukaan uuteen ohjelmaan. Tämä saattaa helposti tuottaa pettymyksiä joillekin osallistujille, mikäli yhtäkään heidän toivettaan ei osata huomioida. Haastattelun tuloksista ei voitu, eikä haluttu yksilöidä asiakkaita, vaan löytää tärkeimmät kohdat.

Ryhmähaastattelussa haasteena on voimakkaat persoonat, jotka saattavat jättää muut ryhmässä olijat varjoonsa. Tämän työn haastattelussa kyseistä ongelmaa pyrittiin välttämään, lisäämällä ryhmiin yrityksen työntekijä auttamaan suunnittelussa. Ryhmien haluttiin saavan aikaiseksi keskusteluja, joissa tulee esiin mielipiteitä ja erilaisia näkemyksiä. Ryhmien kokoamisella on suuri merkitys ryhmähaastattelussa. Laitetaanko kokeneet ja kokemattomat pääkäyttäjät samaan vai eri ryhmiin vai julkisen sektorin yrityksen käyttäjät ja yksityisen sektorin käyttäjät omiin ryhmiin. Kyseisessä työssä haluttiin sekoittaa kaikki alat, kokemusvuodet ja työantajaa katsomatta eri ryhmiin, jotta keskustelulle saataisiin eri lähtökohdat. Ryhmien sekoittaminen saattaa myös aiheuttaa vastakkain asenteita, keskustelua ei synny tai se menee väittelyksi. (Hirsjärvi ja Hurme 2010).

Henkilöhaastattelussa kannattaa suosia avoimia kysymyksiä ja ajautua juttelemaan itsekin niitä näitä (Suhola ym. 2005). Avoimet kysymykset saattavat jättää tärkeät ominaisuudet pois, koska niitä pidetään itsestään selvyytenä. Tutkimuksesta saatu kehityslistan pituus yllätti, minkä vuoksi tärkeimpien ominaisuuksien valitseminen oli haastavaa. Projektiryhmä teki aikaisemman kokemuksen ja asiakaspalautteen perusteella päätökset

valituksi tulleista ominaisuuksista. Laajat kysymykset saatetaan helposti ymmärtää väärin, ja ryhmän keskustelu saattaa lähteä väärään suuntaan. Tämä saattaa tuoda kyseiseltä ryhmältä vääriä määrityksiä sovellukseen tai tarpeiden priorisointia.

Suunnitteluvaiheeseen tuli varata tarpeeksi aikaa, jotta ominaisuuksia voitiin purkaa yhdessä ja pohtia niiden tärkeyttä. Uuden ohjelman projekti olisi voinut epäonnistua heti alkuvaiheessa, jos sopivaa kokonaisuutta ei olisi saatu luotua kaikille tutkimukseen osallistuneille asiakkaille.

Asiakkaiden sitouttaminen koko projektiin oli hankalaa. Kehitysprosessin aikana pyydettiin asiakkaita testaamaan sovellusta jokaisen sprintin jälkeen, kun ohjelman versio oli päivittynyt. Monelta osaa testaaminen jää vähäiseksi ja odotetaan, koska ohjelma voidaan ottaa tuotantokäyttöön. Paremman viestinnän avulla olisi mahdollisesti voitu parantaa testausinnostusta asiakkaissa. Ohjelmalle ei saatu tarpeeksi käyttäjätestaamista ja yrityksen oma testaaminen ei tuonut suorituskykyongelmia esiin datan vähäisyyden vuoksi. Palautetta ei juurikaan tullut ohjelman ominaisuuksista projektin kehitysvaiheessa, vaan asiakkaat olettivat toivomiensa ominaisuuksien löytyvän jo tai myöhemmässä vaiheessa ohjelmasta. Projektin suuri asiakasmäärä tuntui vaikuttavan asiakkaiden testausinnostukseen; moni asiakas oletti varmasti testauksen tapahtuvan jonkun toisen asiakkaan toimesta.

Tuotantokäyttöönoton jälkeen paljastui muutamia ongelmia, jotka hankaloittivat uuden ohjelman täysimääräistä käyttöä. Ongelmat johtuivat suurista datamääristä ja niiden latausajoista. Käyttäjät tuskailivat hitauden kanssa alkuun, mutta ongelmat saatiin nopeasti kiinni ja ohjelmaan lisättiin asetuksia. Asetusten avulla ohjelmaa pystyttiin keventämään ja muokkaamaan jokaisen käyttäjän mukaiseksi. Kaikki turha lataaminen ja data voitiin jättää pois näkyviltä. Ilman hitausongelmia ohjelma toimi erinomaisesti ja nopeutti pääkäyttäjien työtä.

Projektia kehitettiin ketteriä menetelmiä hyväksikäyttäen. Sovelletun Scrumin avulla asiakkaat pystyttiin pitämään lähellä projektia ja tietosina sen etenemisestä. Tarvittaessa ominaisuuksia pystyttiin muuttamaan projektin aikana. Ketterä projektimalli todettiin toimivana ratkaisuaan tässä opinnäytetyössä, koska suurin osa ongelmista ja puutteista saatiin korjattua kehityksen aikana. Oikein valittu menetelmä auttaa projektinhallinnassa ja tekee työskentelystä sujuvampaa, nopeampaa ja tehokkaampaa, välittäen lisäarvoa myös asiakkaille. (Visma 2018)

Työssä haastattelumenetelmä oli käytössä ensimmäistä kertaa, joten olennaista on pohdita, saatiinko haastatelluilta asiakkailta kaikki tarvittava ja riittävä tieto ohjelmiston kehittämiseen. Ryhmähaastattelun luotettavuutta saattaa heikentää se, että haastateltavalla on taipumus antaa sosiaalisesti suotavia vastauksia tai tietoa aiheesta. Yrityksen edustaja pyrki samaan oikeita tuloksia ryhmiltä, mutta vaikuttiko ohjaaja liikaa ryhmän vastauksiin. Reliabiliteetin eli toistettavuuden haasteellisuus on saada saman tasoiset ryhmät myös seuraavassa tutkimushaastattelussa. Tässä tutkimuksessa pyryttiin vahvistamaan reliabiliteettia johdonmukaisuudella ja sisäisen yhtenäisyyden varmistamisella. Haastattelututkimukseen ei saatu kaikkia nykyisiä asiakkaita mukaan, joten ei voitu varmistaa, että ohjelman lopputulos tyydyttää kaikki asiakkaita.

Tulevaisuudessa samaa kehitysmenetelmää tullaan käyttämään yrityksessä jatkossakin. Palaute oli positiivista työntekijöiden ja asiakkaiden puolelta. Tulevaisuudessa asiakkaat tulisi pitää tiiviimmin mukaan kehityksessä ja pitää useampi haastattelutilaisuus. Projektin kehittyessä testausta voitaisiin parantaa samalla tutkimuksella. Keskustelun avulla puutteet ja ongelmat saadaan aikaisemmassa vaiheessa kiinni ja suurimmat puutokset korjattua. Asiakkaat pitivät toimintamallia erittäin onnistuneena ja aikovat itse jatkossa käyttää samaa tapaa. Tietyt ominaisuudet puuttuivat joidenkin asiakkaiden tarpeisiin nähden ensimmäisestä versiosta. He käyttävät sovellusta hybridimallina, eli voivat siirtyä uudesta sovelluksesta suoraan vanhan sovelluksen haluttuun paikkaan. Hybridimalli on tärkeä jatkuvuuden takaamiseksi. Yrityksessä on paljon asiakaskohtaisia räätälöintejä, jolloin sovelluksen uudistaminen ilman hybridimallia on hyvin työlästä. Ohjelman jatkokehitystä varten on suunnitteilla uusi samanlainen tilaisuus. Asiakkaan huomioimien koko projektin ajan sitouttaa asiakkaan käyttämään ohjelmaa jatkossakin. Järjestelmää ei tarvitse sitouttaa muutokseen, mutta ihminen pitää, jotta loppukäyttäjät osaavat toimia uudessa maailmassa. (Linnainmaa 2017)

7 YHTEENVETO

Opinnäytetyössä oli tarkoituksena kehittää ja uudistaa yrityksen projektikehitysmenetelmää, sekä tapahtumahallinnan ohjelma nykyaikaseksi web-sovellukseksi. Tähän lähdettiin hakemaan vastauksia ryhmähaastattelun ja siitä saatujen vastausten perusteella. Tarkoituksena oli saada asiakkaat innostumaan uudella tavalla raskaiden projektien kehittamisestä ja osallistaa heidät suunnittelemaan merkittävä osa uudesta ohjelmasta. Kyseisen toimintamallin avulla pystyttiin säästämään huomattava määrä aikaa projektiryhmältä, verrattuna siihen, että sen olisi tullut suunnitella ohjelman määrittelyt itse.

Tämän opinnäytetyön projekti toteutettiin ketterää kehitysmenetelmää hyväksi käyttäen. Ketterän mallin avulla projekti saatiin ajoissa valmiiksi ja tuotannossa ilmenneet virheet pystyttiin korjaamaan nopealla aikataululla. Asiakkaan ja projektiryhmän oli helppo seurata projektin etenemistä ja uusien ominaisuuksien valmistumista uuteen ohjelmaan.

Opinnäytetyssä toteutettu tapahtumahallinnan ohjelman suunnittelu ja toteutus onnistui kokonaisuudessaan erittäin hyvin. Asiakkaat olivat lopputulokseen tyytyväisiä ja valmiita jatkamaan jatkokehitysprojektissa. Yrityksessä ryhmähaastattelumallia pidettiin hyvänä ja nopeana tapana kehittää ohjelmaa. Yritys ja asiakkaat saivat kyseistä työstä paljon kehitysideoita ja tietoa tulevia projekteja varten.

LÄHTEET

Aden D.; Aden J. & Wilken J. 2016. *Angular 2 in Action*. Manning Publications.

Agile Alliance. 2001. *Manifesto for Agile Software Development*. viitattu 12.3.2021. <https://agilemanifesto.org/>

Arora C. & Hennessy K. 2018. *Angular 6 eexample*. 3. painos. Packt

Blomqvist H. 2019. *Miten varmistan projektin epäonnistumisen – lue 10 vinkkiä*. viitattu 19.4.2021 <https://blog.oppia.fi/2019/09/27/miten-varmistan-projektin-epaonnistumisen-lue-10-vinkkia/>

Boduch A. 2016. *Flux Architecture*. 1. painos. Packt

Developer chrome. 2014. *MVC Architecture* https://developer.chrome.com/apps/app_frameworks, 2014

Elo M. 2014. *Viestintä IT-projekteissa: merkitys, hallinta ja esteet*. Jyväskylän yliopisto Pro Gradu, s.71 <http://urn.fi/URN:NBN:fi:juu-201412103472>

Eloranta J. 2020. *SPA-sovellukset hajautettuna järjestelmänä*. Matemaattis-luonnontieteellinen tiedekunta Helsingin yliopisto. s. 65 <http://urn.fi/URN:NBN:fi:hulib-202005202224>

Engberg D. 2021. *How To Use the Agile SCRUM Framework for IT Infrastructure*. viitattu 5.5.2021 <https://www.agdiwo.com/en/agile-infrastructure/>

Exbrayat C. 2018. *Become a ninja with Angular 2*. 3. painos. Ninja squad

Fain Y. & Moiseev A. 2019. *Angular 2 Development with TypeScript*. 2. painos. Manning Publications

Farrell B. 2019. *Web Components in Action*. 1. painos. Manning Publications

Fink G. & Flatow I. 2014. *Pro Single Page Application Development. Using Backbone.js and ASP.NET*. Apress.

Haikala I. & Mikkonen T. 2011. *Ohjelmistotuotannon käytännöt*. Talentum

Hirsjärvi S & Hurme H. 2015. *Tutkimushaastattelu: Teemahaastattelun teoria ja käytäntö*. 2. painos. Gaudeamus

Hyvärinen M. & Ruusuvuori J. 2017. *Tutkimushaastattelun käsikirja*. Osuuskunta Vastapaino

Kokkonen, J. 2015. *Single-page Application Frameworks in Enterprise Software Development* (YAMK). Jyväskylän ammattikorkeakoulu. s. 44 <http://urn.fi/URN:NBN:fi:amk-2015060912877>

Korsström F. 2013. *Mistä tunnistaa onnistuneen IT-projektin?*. Viitattu 12.4.2021. <https://www.talouselama.fi/kumppaniblogit/mista-tunnistaa-onnistuneen-it-projektin/47df0b1c-b898-3adc-82df-948ed7d10158>

Laitinen S. 2020. *Projektien onnistuminen IT-alalla*. Tampereen yliopisto. kandidaattityö. s. 26 <http://urn.fi/URN:NBN:fi:tuni-202005054963>

Lawson K. 2018. *What Is a Single Page Application and Why Do People Like Them so Much?*. viitattu 19.4.2021 <https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html>

Lekman L. 2010. *Vesiputouksen alkulähteillä*. viitattu 5.5.2021 <https://lekman.fi/2010/05/31/vesiputouksen-alkulahteilla/>

Leppänen T. 2020. *Mihin tarvitaan ei-toiminnallisia IT-vaatimuksia?*. viitattu 20.4.2021 <https://www.cheetah.fi/blog/?p=259>

Lind O. 2001 *Näin tehdään onnistunut projekti*. Domus Offset Oy Tampere

Linnainmaa N. 2017. *Miksi projekteissa kannattaa käyttää asiantuntijoiden apua?*. viitattu 3.5.2021 <https://www.project-it.fi/2017/06/07/projektijohtamisen-abc-osa-2-miksi-projekteissa-kannattaa-kayttaa-asiantuntijoiden-apua-lue-vinkit/>

Mäntyneva M. 2016. *Hallittu projekti: järkevästä suunnittelusta menestykselliseen toteutukseen*. 2. painos. Kauppakamari

Mozilla Developer Network. 2020. *Web Components*. Viitattu 09.09.2020 WWW-muodossa. https://developer.mozilla.org/en-US/docs/Web/Web_Components/HTML_Imports

Mtech digital solutions 2019. *Onnistuneen IT-projektin avaimet*. viitattu 1.5.2021 <http://urn.fi/URN:NBN:fi:amk-2015052610416>

Nieuwenhuis S. 2019. *The Ultimate Guide to Web Components*. Viitattu 09.09.2020 <https://ultimatecourses.com/blog/the-ultimate-guide-to-web-components#what-are-web-components>

Ruuska K. 2001. *Projekti hallintaan*. Gummerrus Jyväskylä

Schwaber K. & Sutherland J. 2020. *Scrum-opas*. viitattu 4.5.2021 <https://scrum-well.files.wordpress.com/2021/02/2020-scrum-guide-finnish.pdf>

Suhola A.; Turunen S. & Varis M. 2005. *Journalistisen kirjoittamisen perusteet*. Helsinki: Oy Finn Lectura.

Thinkingportfolio. 2016. *Kuinka valita sopiva menetelmä projektiin?*. viitattu 16.4.2021 <https://thinkingportfolio.com/kuinka-valita-sopiva-menetelma-projektiin>

Törnqvist P. 2015. *Ketterät menetelmät: Muutos vesiputousmallista ketteriin menetelmiin siirryttäessä kansainvälisissä ohjelmistokehitysorganisaatioissa*. Laurea-ammattikorkeakoulu. s. 45 <http://urn.fi/URN:NBN:fi:amk-2015052610416>

Vanhala L. 2012. *Näin VR sotki lippujärjestelmänsä – Miksi it-projektit epäonnistuvat?*. viitattu 21.4.2021. <https://suomenkuvalehti.fi/jutut/kotimaa/nain-vr-sotki-lippujarjestelmansa-miksi-it-projektit-epaonnistuvat/>

Virginia tech. 2018. *Research Methods Guide: Interview Research*. viitattu 21.4.2021 <https://guides.lib.vt.edu/researchmethods/interviews>

Visma Solutions. 2018. *Ketterät menetelmät projektinhallintaan*. viitattu 3.5.2021
<https://psa.visma.fi/blog/ketterat-menetelmat-projektinhallinta/>

webcomponents.org 2021. *What are web components?*. viitattu 21.4.2021
<https://www.webcomponents.org/introduction>

Wilken J. 2018. *Angular in Action*. 1. painos. Manning Publications.