Koulutuksen suunnittelutyökalu

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja viestintätekniikka
2022
Niko Koistinen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Koistinen, Niko	Opinnäytetyö, AMK	Kevät 2022
	Sivumäärä	
	39+2	

Työn nimi

Koulutuksen suunnittelutyökalu

Tutkinto

Insinööri (AMK), Tieto- ja viestintätekniikka

Toimeksiantajan nimi, titteli ja organisaatio

Puolustusvoimat, Utin jääkärirykmentti

Tiivistelmä

Opinnäytetyössä toteutettiin tietokantapohjainen suunnittelusovellus Laskuvarjojääkärikomppanian käyttöön. Sovelluksen tavoitteena oli helpottaa koulutuksen suunnittelua sekä tehostaa koulutuksen seurantaa, kehittämistä ja raportointia.

Opinnäytetyöhön kerättiin monipuolisista lähteistä ohjelmistokehityksen ja projektinhallinnan hyviä käytänteitä, joita sovellettiin tämän työn toteuttamisessa. Sovellus toteutettiin Access -tietokantatyökalun avulla. Sovellus otettiin testikäyttöön suunnitellussa aikataulussa, mutta osa sovellukselle asetetuista tavoitteista jäi vielä toteuttamatta. Käyttöönotto vaiheistettiin uudelleen niin, että sovellus korvaisi nykyjärjestelmän vuoden 2023 aikana.

Sovellus antaa mahdollisuuden käsitellä suunnittelutyön tuotteita paremmin, minkä johdosta koulutusten suunnitteluun kuluvaa työmäärää kyetään tulevaisuudessa vähentämään. Jatkokehitysvaiheessa sovellukseen toteutetaan loput sille asetetuista tavoitteista ja sovelluksen kehittämistä jatketaan käyttäjien kehitysehdotusten mukaisesti osana työyksikön normaalia hallintoa.

Asiasanat

ohjelmisto, ohjelmistokehitys, sovellus, tietokannat

Abstract

Author(s)	Type of Publication	Published
Koistinen, Niko	Thesis, UAS	Spring 2022
	Number of Pages	
	39+2	

Title of Publication

Training planning tool

Name of Degree

Engineer (UAS), Bachelor's Degree Programme in Information Technology

Name, title and organization of the client

Finnish Defence Forces, Utti Jaeger regiment

Abstract

The aim of this thesis was to create an application for the Paratrooper company to facilitate and improve the planning, monitoring, development and reporting of training. For the thesis, good software development and project management practices were collected from various sources and applied in the implementation of this work.

The application was deployed into test use according to the planned schedule, but some of the objectives were not achieved. The deployment of the application was rephased so that the application would replace the current system during 2023. The application provides more efficient ways to handle planning products, reducing time and effort that goes into planning.

In the further development phase, the remaining goals will be achieved, and application development is continued in accordance with user development suggestions as part of normal work unit management.

Keywords

software, software development, database

Sisällys

Liitteet

1 Johdanto

Opinnäytetyön tavoitteena on tehdä tietokantasovellus Laskuvarjojääkärikomppanian suunnittelutyön tueksi. Opinnäytetyössä toteutetaan kirjallisuuskatsaus ohjelmistokehityksen näkökulmasta merkittävimpiin tietolähteisiin ja käytäntöihin, joita pyritään soveltamaan lopputyön toteuttamisessa.

Sovelluksen tavoitteena on helpottaa yksittäisten koulutusten suunnittelua ja valmistelua, viikko-ohjelmien luontia sekä henkilöresurssien tasapuolista ja tehokasta käyttöä. Sovelluksella pyritään myös ratkaisemaan nykyjärjestelmässä havaittuja ongelmia kuten historiatietojen hankala selattavuus, suunnittelusta syntyvien tuotteiden pirstaleisuus sekä, osin näistäkin johtuvat, harjoitusten kehittämiseen liittyvien toimien puutteellisuus. Myös kokemusperäisen tiedon menettämistä henkilöstövaihdosten yhteydessä pyritään minimoimaan.

Sovelluksen kehittämistä varten suoritetaan alkukartoitus nykytilanteesta ja kerätään suunnittelutyöhön osallistuneilta heidän näkemyksensä ratkaistavista ongelmista. Sovelluksen toteuttamiseen luodaan erillinen projektisuunnitelma, missä pyritään hyödyntämään hyviä projektinhallinnan käytänteitä.

Työssä ei oteta kantaa yksikön hallinnon toteuttamiseen tai sen kehittämiseen vaan työllä pyritään poistamaan nykyisen järjestelmän aiheuttamia puutteita ja kehittää toimintaa sitä kautta. Jos työn edetessä kuitenkin havaitaan selkeitä toimintatapamalleja, joita muuttamalla voitaisiin osa ongelmista ratkaista tai jos niiden mahdollinen käyttöönotto vaikuttaisi sovelluksen toteuttamiseen, otetaan ne osaksi lopputyötä ja toimenpidemuutosesitykset esitellään tilaajalle projektiin liittyvissä ohjauspalavereissa.

2 Organisaation esittely

2.1 Puolustusvoimien erikoisjoukot

Puolustusvoimat turvaa Suomen aluetta, kansan elinmahdollisuuksia ja valtiojohdon toimintavapautta sekä puolustaa laillista yhteiskuntajärjestystä tarvittaessa sotilaallisin voimakeinoin aseellisen hyökkäyksen tai sitä vastaavan ulkoisen uhan kohdistuessa Suomeen. Puolustusvoimilla on neljä lakisääteistä tehtävää:

- Suomen sotilaallinen puolustaminen
- Muiden viranomaisten tukeminen
- Osallistuminen kansainvälisen avun antamiseen
- Osallistuminen kansainväliseen sotilaalliseen kriisinhallintaan

Rauhanaikana Puolustusvoimilla on palveluksessaan noin 12000 palkattua henkilökuntaan kuuluvaa ja varusmiehiä koulutetaan sodanajan tehtäviin vuosittain noin 22000. (Puolustusvoimat 2021.)

Puolustusvoimien hakuprosessissa erikoisjoukoilla tarkoitetaan erillishaussa haettavia joukkoja, joihin kuuluu mm. elektronisen sodankäynnin joukot, erikoisrajajääkärit, hävittäjä- ja apumekaanikot sekä varusmiessoittajat (Intti.fi 2022). Erikoisjoukot voidaan siis erotella pelkästään sillä, että niihin ei määrätä vaan niihin hakeudutaan vapaaehtoisesti. Yleisesti erikoisjoukolla kuitenkin tarkoitetaan osastoa, joka on koulutettu ja varustettu tavanomaisista sotilasoperaatioista poikkeaviin tehtäviin. Näille erikoisjoukoille on tyypillistä, että ne koostuvat fyysisesti ja henkisesti kestävistä sotilaista, jotka kykenevät toimimaan haastavissa erikoisjoukko-operaatioissa kaikissa olosuhteissa.

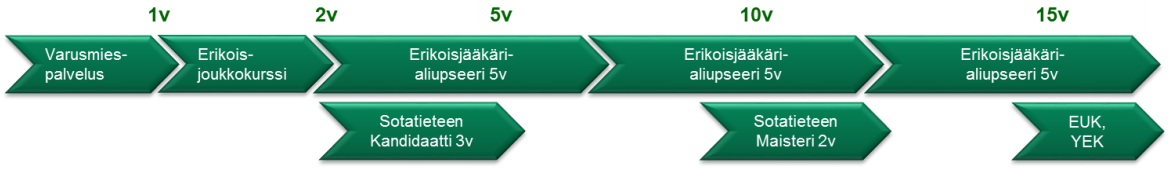
Erikoisjääkäripataljoonan oman esittelymateriaalin (2016) mukaan erikoisjoukko-operaatiot ovat tehtävätyypeiltään, olosuhteiltaan, toteutustavoiltaan ja vaikuttavuuksiltaan hyvin moninaisia. Käytännössä nämä yleensä esiintyvät fyysisesti, henkisesti sekä osaamistasoltaan erittäin haastavina tehtävinä, mihin liittyy usein monien siirtymismenetelmien hyödyntäminen. Siirtymismenetelmät voivat vaihdella jalka-/suksipartioinnista moottorikelkkoihin tai jopa lentokoneisiin ja helikoptereihin.

Laskuvarjojääkäreiden osalta tehtävät keskittyvät vihollisen syvyyteen, jossa olosuhteet, omavaraisuus ja pitkät siirtymiset luovat henkisesti ja fyysisesti raskaan ympäristön. Erikoisjääkäreillä on syvyystehtävien lisäksi kyky toteuttaa haastavia vastaerikoisjoukko-operaatioita omalla alueella sekä merellä.

Puolustusvoimien erikoisjoukkojen koulutus alkaa aina vuoden kestävällä varusmiespalveluksella, joista laskuvarjojääkäri on yksi koulutushaara. Varusmiespalveluksen ei ole kuitenkaan tarvinnut kohdistua erikoisjoukkoihin, sillä missä tahansa suoritettu reservin aliupseerin tai upseerin koulutus oikeuttaa hakeutumaan Puolustusvoimien erikoisjoukkokurssille (Kuvio 1).

Puolustusvoimien erikoisjoukkokurssin englanninkielinen nimi on qualification course, mikä kuvaa osuvasti sen tarkoitusta: erikoisjoukkokurssilla opetetaan erikoisjoukkojen perustoimintatapamallit ja harjaannutaan pienryhmätekniikoihin ja -taktiikoihin, joilla luodaan pätevyys hakeutua erikoisjääkärin ammattiin. Tällä hetkellä ainoa tie erikoisjääkäriammattisotilaaksi (erikoisjääkärialiupseeri) on suorittaa hyväksytysti Puolustusvoimien erikoisjoukkokurssi.

Erikoisjääkärialiupseerit palvelevat Erikoisjääkäripataljoonan Erikoisjääkärikomppaniassa. Erikoisjääkärialiupseerin koulutusjärjestelmä on tällä hetkellä 3-osainen, joista kukin palvellaan 5 vuoden määräaikaisessa aliupseerin virassa. Näiden 15 vuoden aikana aliupseerit harjaantuvat erikoisjoukkojen tehtävissä, joihin kuuluu kansallisen puolustuksen lisäksi viranomaisyhteistyö sekä kansainväliset kriisinhallintatehtävät.



Kuvio 1. Erikoisjoukkojen koulutusjärjestelmä (Erikoisjääkäripataljoona 2016)

Erikoisjoukkojen upseeritehtäviin hakeudutaan joko suoraan Maanpuolustuskorkeakoulusta Laskuvarjojääkärikomppaniaan, jossa perehdytään erikoisjoukkojen tehtävätyyppeihin tai sisäisessä haussa vapaana olevaan paikkaan Laskuvarjojääkärikomppaniassa. Laskuvarjojääkärikomppaniassa palvelleet upseerit voidaan rekrytoida Erikoisjääkärikomppaniaan, jossa heidät koulutetaan toimimaan suunnittelu- ja johtotehtävissä erikoisjoukkooperaatioissa. Tyypillisesti Erikoisjääkärikomppaniaan rekrytoitu upseeri on joko palvellut varusmiehenä Laskuvarjojääkärikomppaniassa, suorittanut Puolustusvoimien erikoisjoukkokurssin tai molemmat, ennen upseeriopintojen aloittamista Maanpuolustuskorkeakoulussa. Tällaisessa tapauksessa siirtyminen Erikoisjääkärikomppaniaan on tietenkin luonteva ja pohja jatkokoulutukseen on hyvä. Oli siirtotapa tai -aikataulu mikä tahansa, toteutetaan upseerin urakehitykseen liittyvät maisteri- ja yleisesiupseeri- tai esiupseerikurssit normaalin suunnitelman mukaisesti. Opinnäytetyössä esiintyvä termi perusyksikkö tai yksikkö tarkoittaa myös komppaniaa tai sen kokoista yksikköä.

Erikoisjoukoille pidetään tavanomaisista joukoista poikkeavaa koulutusta, joten kussakin tasossa joudutaan suunnittelemaan, toimeenpanemaan ja kehittämään koulutusta omien tarpeiden mukaan. Tämän takia myös perusyksikön kouluttajan arkeen kuuluu painokkaasti suunnittelutehtävät, joita toteutetaan päätehtävän ohessa.

2.2 Työyksikön hallinto

Työyksikön hallintoa koskevissa asioissa on käytetty lähteinä Utin jääkärirykmentin Koulutusosaston sekä Laskuvarjojääkärikomppanian omia esittelymateriaaleja. Laskuvarjojääkärikomppaniassa koulutusta ohjaavat opetus- ja koulutussuunnitelmat sekä varusmieskoulutukseen liittyvät normit. Niissä käsketään komppanialle vaatimukset joukkotuotantoon. Näiden vaatimusten perusteella komppanian johto suunnittelee vuosigrafiikan, koulutusjaksojen läpiviennit, viikko-ohjelmat ja tuntisuunnitelmat. Laskuvarjojääkärikomppanian henkilökuntaan kuuluvat kouluttajat vastaavat läpivientien ja viikko-ohjelmien pohjalta yksittäisten harjoitusten suunnittelusta, toteuttamisesta ja raportoinnista.

Erikoisjoukkojen toimintaan on ominaista, että joukolle asetetut vaatimukset muuttuvat tiheämmällä syklillä kuin tavanomaisten joukkojen. Tähän vaikuttavat muun muassa teknologian kehitys ja turvallisuuspoliittiset muutokset. Silti arviolta 90 % kaikesta koulutuksesta on lähestulkoon identtistä jokaisella vuosikurssilla.

Kaikkiin työyksikön työntekijöihin sovelletaan virkaehtosopimusta ja työaika on jaksotyöaika. Tämä tarkoittaa sitä, että työajat suunnitellaan 3 viikon jaksoissa, jossa kunkin jakson säännöllinen työaika on 114 h 45min (7h 39min/pv). Säännöllistä työaikaa on pääosa kalenterivuoden työpäivistä, mutta sotilaat on mahdollista käskeä myös sotilaallisiin harjoituksiin. Näinä päivinä työskentelyaika voi olla 24 h / vrk. Jos työntekijöille aiheutuu säännöllistä työaikaa pidempiä työpäiviä, tulee nämä tunnit tasoittaa kuluvan jakson aikana. Tämä voi tarkoittaa, että työntekijällä on joko yksi tai useampi lyhyt työpäivä tai vapaapäiviä.

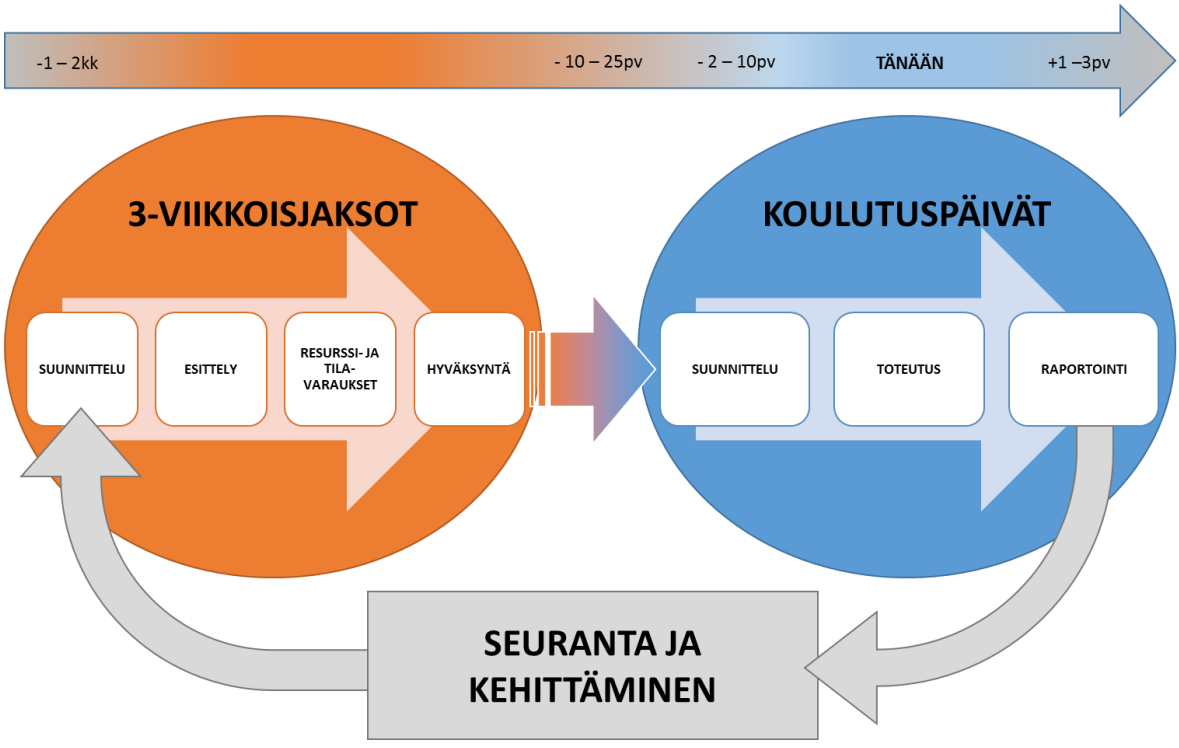
Työyksikön suunnittelu on monivaiheista. Vuosisuunnittelua tehdään yhteistoiminnassa esikuntien ja muiden tukevien elementtien kanssa muutaman kerran vuodessa tapahtuvilla suunnittelutilaisuuksilla. Suunnittelutilaisuuksien päätteeksi syntyy vuosigrafiikka, missä päätetään viikkotarkkuudella koko seuraavan vuoden läpivienti.

Jaksosuunnittelua toteutetaan johdon toimesta jatkuvasti ja suunnittelu on joustavaa: kun akuutteja työtehtäviä on vähemmän, suunnitteluun mahdollistuu aikaa enemmän. Selkein jaksosuunnittelua ohjaava sopimusosa on se, että jokaisen työntekijän on saatava tietää seuraavan jakson työvuoronsa käynnissä olevan työjakson keskimmäisen viikon keskiviikkona (1,5 viikkoa ennen tulevan jakson alkua). Seuraavan työjakson esittely onkin suunniteltu sitä edeltävälle päivälle, jolloin ehditään tehdä viimeiset muutokset ja korjaukset ennen

jaksosuunnitelman hyväksyntää. Esittelyä ja hyväksyntää ei aikaisteta siitä syystä, että mitä aikaisemmin tulevan jakson työsuunnitelmat ja tehtävät on lyöty lukkoon, sitä enemmän muutoksia siihen jaksoon on jouduttu tekemään. Siitä huolimatta hyväksytyn jakson jalkauttamisessa on työntekijöille esitelty myös karkea luonnos sitä seuraavasta jaksosta, jotta heillä on parempi mahdollisuus suunnitella myös siviilielämäänsä. Jaksosuunnittelun tuotoksena syntyy tuntisuunnitelma ja viikko-ohjelmat, joista selviää ohjelma ja vastuuhenkilöt päivän tarkkuudella.

Yksittäisten harjoitusten suunnittelu, toteutus ja raportointi ovat suunnitelmaan siihen nimetyn vastuuhenkilön vastuulla. Yksi koulutuspäivä pitää sisällään normaalisti yhdestä neljään erillistä harjoitusta aiheesta riippuen. Kestot siis vaihtelevat noin 30 minuutista 8 tuntiin. Nämä harjoitussuunnitelmat, tai joissain tapauksissa voidaan puhua jopa koulutuskorteista, pitävät sisällään yksityiskohtaiset ohjeet ja tavoitteet sekä suunnitelman ajankäytöstä minuutin tarkkuudella. Koulutustapahtumista pyritään keräämään huomiot, mitä voidaan käyttää tulevien jaksojen suunnitteluun sekä opetus- ja koulutussuunnitelmien päivittämiseen.

Jaksosuunnittelun ja koulutuspäivien suunnittelusykli on esitetty kuviossa 2.



Kuvio 2. Jaksosuunnittelun ja koulutuspäivien suunnittelun sykli (Laskuvarjojääkärikomppania 2022)

Nykytilanteen esittely

Tällä hetkellä henkilöstön työsuunnitelmien ja viikko-ohjelmien tekemiseen käytetään niiden suunnitteluun ja taltiointiin rakennettuja, määrämuotoisia taulukkolaskentatiedostoja, joista

on esitelty esimerkit liitteessä 1. Nämä tuotteet muodostavat siis parin, jossa toisessa kerrotaan, mitä tehdään (viikko-ohjelma) ja toisessa varataan henkilöresurssit sen tekemiseen (tuntisuunnitelma). Tämä järjestelmä on ollut käytössä jo vuosikymmeniä, minkä aikana se on kokenut vain kaksi isompaa muutosta: ensin tietokoneiden yleistyessä kirjoituskoneet vaihdettiin tekstinkäsittelyohjelmiin ja 2017 tekstitiedostot muutettiin taulukkolaskentamuotoon. Vuosien varrella tuotteiden suunnittelu ja käytännön toteutus on siis helpottunut, mutta taltiointi ja seuranta tuottavat edelleen ongelmia: pelkästään tuntisuunnitelma- ja viikko-ohjelmatiedostoja syntyy vuosittain vähän yli 100 kappaletta. Esimerkiksi vastaukset kysymyksiin "onko pidetty koulutus aiheesta x?" tai "kuinka monta tuntia työntekijä y on pitänyt koulutuksia tänä vuonna?" löytyvät ainoastaan selaamalla näitä tiedostoja läpi yksi kerrallaan.

Koko vuoden hahmottelua helpottaakseen suunnittelusta vastaavat henkilöt käyttävät lisäksi kahta taulukkolaskentatiedostoa. Toisessa tiedostossa hahmotellaan kaikkien työntekijöiden menot ja paikallaolot koko vuoden osalta (henkilöstösuunnittelu). Toisessa näkyy kaikki vuoden koulutuspäivät, mihin voidaan otsikkotarkkuudella suunnitella karkea teema viikoittain tai päivittäin (koulutussuunnittelu).

Toimintaa kehitetään henkilöstön huomioiden ja palautteiden mukaan. Tällä hetkellä huomioiden kerääminen tapahtuu pääsääntöisesti aamupalaverissa, missä edellisen päivän vastuuhenkilöt esittelevät päivän toteutuksen ja omat havaintonsa, miten kyseisiä harjoituksia voisi kehittää tulevaisuudessa. Raportointi voi jäädä usein puutteelliseksi, jos edellisen päivän vastuuhenkilö tai jaksosuunnittelusta vastaavat henkilöt ovat poissa työpisteeltä esimerkiksi työmatkan tai tasoitusvapaiden takia. Tämän takia raportointi on ajautunut käsittelemään lähinnä sitä, onko aika ollut riittävää, pitääkö kyseinen aihe kouluttaa vielä uudelleen ja suositukset, miten aiheen syventäminen kannattaa seuraavassa koulutustilanteessa toteuttaa.

Yksittäisten harjoitusten harjoitussuunnitelmat toteutetaan tällä hetkellä melko vapaamuotoisesti. Jos suunnitteluaikaa on riittävästi, harjoituksesta muodostetaan koulutuskortti, mikä on määrämuotoinen suunnittelutyökalu. Jos suunnittelu- ja valmistautumisaikaa ei ole ollut riittävästi, harjoitussuunnitelmaan kirjataan omiin muistiinpanoihin vain päätavoitteet, muutama huomio ja hahmotelma ajankäytöstä.

3 Ohjelmiston elinkaari

3.1 Vaihejakomallit

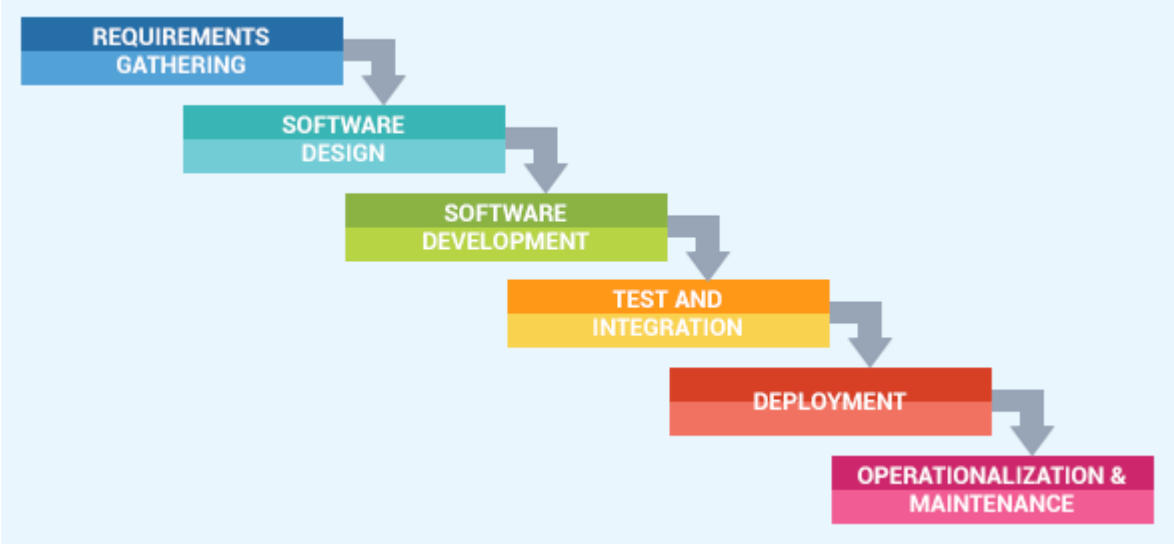
Ei ole yhtä oikeaa tapaa toteuttaa sovellusta, mutta ohjelmistokehitystyössä on havaittu vuosikymmenien aikana hyviä käytänteitä ja selkeitä sudenkuoppia. Ohjelmiston elinkaarella tarkoitetaan kaikkia vaiheita ja toimenpiteitä aina alkukartoituksesta ohjelmiston käytöstä poistamiseen saakka. Elinkaaren vaihejakomallista riippuen vaiheita voi olla aina muutamasta kymmeneen. Vaikka näissä malleissa esiintyy eri määrä vaiheita, on niistä havaittavissa kuitenkin selkeitä yhtymäkohtia, joita ovat vaatimusten määrittely, ohjelmiston suunnittelu, ohjelmiston toteuttaminen ja käyttöönotto (Taulukko 1). Ohjelmistokehityksen jakaminen elinkaarimallin vaiheisiin helpottaa osapuolia hahmottamaan ohjelmistokehityksen kokonaisuuden, parantaa tilaajan mahdollisuuksia osallistua projektiin, minimoi riskejä muun muassa kustannusten tai aikataulun osilta sekä vaiheistus mahdollistaa selkeät välietapit, jossa on mahdollisuus tarkastaa ohjelmiston vastaavuuden annettuihin kriteereihin. (Demchenko 2021.)

5 vaihetta (Stiner, 2016)	6 vaihetta (Harness, 2021)	7 vaihetta (Preston, 2021)	10 vaihetta (US DoJ, 2003)
Research	Requirements gathering	Planning	Initiation
Ideation	Software design	Analysis	System Concept Development
Design	Software development	Design	Planning
Development	Test & Integration	Development	Requirements Analysis
Iteration	Deployment	Testing	Design
	Operationalization & Maintenance	Integration	Development
		Maintenance	Integration and Test
			Implementation
			Operations and Maintenance
			Disposition

Taulukko 1. Ohjelmiston elinkaaren vaihejakomalleja

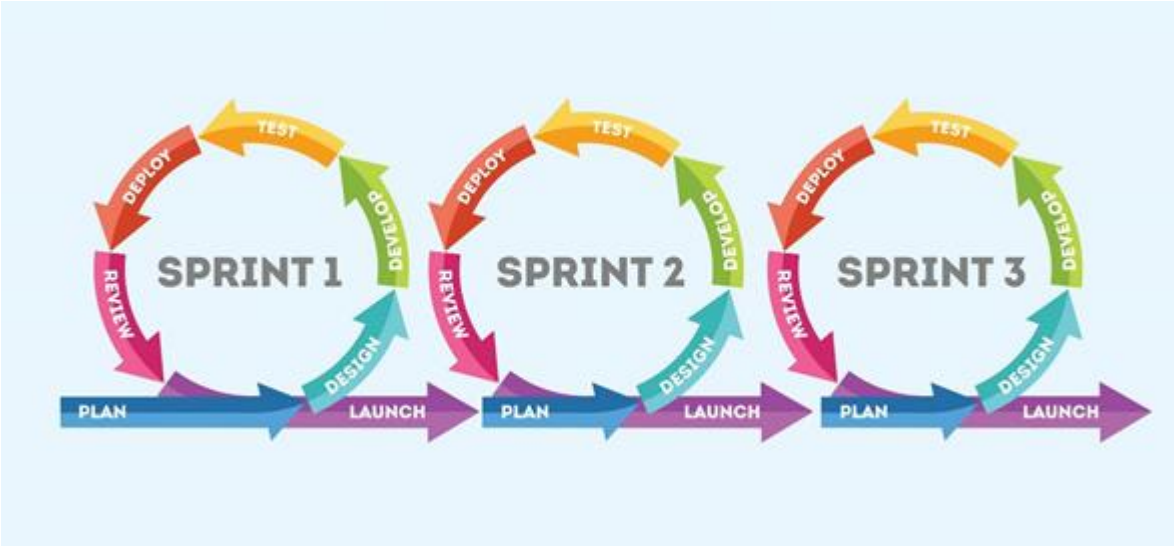
Elinkaarimallin vaiheita voidaan myös toteuttaa eri tavoilla. Ohjelmistokehityksessä käytetyimmät tavat ovat vesiputousmalli ja erilaiset ketterät menetelmät (Majewski, 2019).

Vesiputousmallille on ominaista se, että kukin vaihe toteutetaan alusta loppuun saakka ennen siirtymistä seuraavaan vaiheeseen ja tavoitteena onkin, ettei edelliseen malliin enää palata (Kuvio 3). Tällaisessa lähestymistavassa selkeitä etuja ovat aikataulujen ja kustannusten pitäminen, kattavan suunnittelun jälkeen muutokset ovat hyvin maltillisia, suoraviivainen malli on kaikkien osapuolien helppo ymmärtää ja hahmottaa sekä dokumentointi on usein niin tarkkaa, että henkilöstön vaihtuminen kesken projektin ei aiheuta suuria ongelmia. Vesiputousmalli voi olla hyvä valinta, jos asiakkaalla ei ole laajaa osaamista puuttua ohjelmiston kehittämiseen tai jos ohjelmistokehitysprojekti on riittävän tarkkarajainen ja yksinkertainen. Haastavinta mallissa on se, että kattavimmassakin suunnitelmassa voi olla puutteita. Näihin projektin aikana havaittuihin puutteisiin, ongelmiin tai muutoksiin on hankala puuttua rikkomatta vesiputousmallin perimmäistä tarkoitusta eli jo päätettyyn vaiheeseen ei enää palata. (Gallagher, Dunleavy & Reeves 2019.)



Kuvio 3. Vesiputousmalli

Ketterässä kehitystavassa voidaan sen sijaan palata useita kertoja suunnittelu-, toteutus-, testaus- ja käyttöönottovaiheisiin (Kuvio 4). Tässä tapauksessa käyttöönotolla ei tarkoiteta järjestelmän luovuttamista asiakkaan käyttöön vaan vaihetta voisi tarkemmin kuvata esittelyvaiheena (demo). Ketterän kehitystavan mukaisia runkoja on monenlaisia, joista käytetyin on scrum (Sacolick, 2020, 1). Siinä tällaisia vaiheiden läpi meneviä kierroksia kutsutaan iteraatioiksi tai sprinteiksi ja niiden kesto on muotoutunut yleensä 1–4 viikon mittaisiksi. Sprintin lopussa esitellään asiakkaalle projektin tila, edellisen sprintin aikaansaannokset ja sovitaan seuraavan sprintin tavoitteet ja aikataulu. (Ammouri 2015.)



Kuvio 4. Ketterä kehitysmenetelmä (Slawek-Polczynska 2020)

Ketterän kehitystavan etuja ovat joustavuus ja sopeutumiskyky. Kun iso ohjelmistoprojekti jaetaan pienempiin komponentteihin ja tehtäviin, on helpompi puuttua projektin aikana havaittuihin puutteisiin ja ongelmiin. Myös asiakkaan osallistuminen ohjelmistokehitykseen riittävän tihein välein auttaa välttämään turhaa työtä, kun ohjausta ja palautetta saadaan jokaisen sprintin päätteeksi. (Sacolick 2020, 2.)

Ketterä kehitystapa ei sovellu käytettäväksi, jos asiakkaalla ei ole aikaa tai riittävästi tietotaitoa osallistua kehitystyöhön tai haluttu lopputulos on mahdollista tuottaa suoraan niin sanotusti kaupan hyllyltä. Jos asiakkaan tavoitteet ovat alusta asti selkeät ja niihin on ennakoitavissa vain hyvin vähän, tai ei ollenkaan, muutoksia, kannattaa kehitystapana käyttää esimerkiksi vesiputousmallia. (Rigby, Sutherland & Takeuchi 2016.)

3.2 Vaatimusmäärittely ja käyttöympäristö

Julkisen hallinnon ja siihen liittyvien hankintojen osalta on luotu julkisen hallinnon suositukset (JHS) -ohjenuora yhdenmukaisten palveluiden tai prosessien kehittämiseen. Suositukset hyväksyttiin julkisen hallinnon tietohallinnon neuvottelukunta (JUHTA), jossa toimi asiantuntijoista koostettu JHS-jaosto vuosina 2016–2019. JHS-järjestelmä lakkautettiin 1.1.2020 tiedonhallintalain voimaantulon myötä, mutta siihen asti kerättyjä ja hyväksyttyjä suosituksia voidaan edelleen hyödyntää. (Suomidigi 2021.)

JHS:n suosituksen (JHS 173 2018, 3; JHS 173, 8) mukaan

> *”Tietojärjestelmän vaatimusten määrittely ja sen laadukas organisointi on onnistuneen tietojärjestelmän hankinnan perusedellytys. Vaatimusten määrittely on vaativaa,*

mutta se säästää projektin kuluissa, nopeuttaa hankkeen läpivientiä ja varmistaa vaadittujen ominaisuuksien tuottamisen”

ja

”Riittämätön vaatimusten määrittely on yleisin yksittäinen syy ohjelmistoprojektien epäonnistumiseen. Joidenkin tutkimusten mukaan vaatimusten määrittely on puutteellinen yli 75 prosentissa kaikista epäonnistuneista projekteista”.

Vaatimusten määrittelyllä tarkoitetaan siis prosessia, minkä tavoitteena on selvittää ohjelmistolle asetettavat vaatimukset sellaisella tarkkuudella, että niiden perusteella voidaan kommunikoida ohjelmistoprojektin eri osapuolille, minkälainen ohjelmistosta halutaan tulevan. Hyvän vaatimuksen tunnusmerkkejä ovat (JHS 173, 6; JHS 173, 20):

- oikeellisuus (tietojärjestelmä täyttää asiakastarpeet)
- yksiselitteisyys (ymmärrettävä ja ymmärretään yhteisellä tavalla)
- täydellisyys (kaikki oleellinen on kuvattu)
- yhdenmukaisuus (ristiriidaton)
- todennettavissa oleva
- laitettavissa järjestykseen (tärkeimmät toiminnot ”ylimpänä”)
- muutettavuus (muutos helppo ja turvallinen)
- jäljitettävyys (osiin voidaan palata ja viitata)
- toteutettavuus (vaatimus on projektin resursseilla tai muilla rajoitteilla mahdollinen toteuttaa)
- mitattavuus (vaatimuksen toteutuminen voidaan jälkikäteen todentaa)

Vaatimukset luodaan, täsmennetään, analysoidaan, priorisoidaan ja hyväksytään, minkä jälkeen niistä laaditaan vaatimusluettelo. Vaatimukset erotellaan toiminnallisiin ja ei-toiminnallisiin vaatimuksiin: toiminnallinen vaatimus kuvaa järjestelmän käyttäytymistä, esimerkiksi syötettäessä tietoa siihen, ja ei-toiminnallinen vaatimus pitää sisällään ne kaikki järjestelmälliset reunaehdot, jotta toiminnalliset vaatimukset on mahdollista toteuttaa (esimerkiksi vasteajat ja käytettävät koodikielet). (JHS 173, 4–6; JHS 173, 24.)

Toiminnallisista vaatimuksista luodaan käyttötapausmalli, mistä selviää käyttäjäroolit, käyttötapauskaaviot ja käyttötilanteet, elementtien väliset suhteet ja yleiset toiminnalliset vaatimukset (kuten määrämuotoisten syöttötietojen tarkistukset). Käyttötapauksista voidaan muodostaa lomakkeita, jossa kuvataan yksityiskohtaisesti eri tilanteiden vaiheet (Taulukko 2).

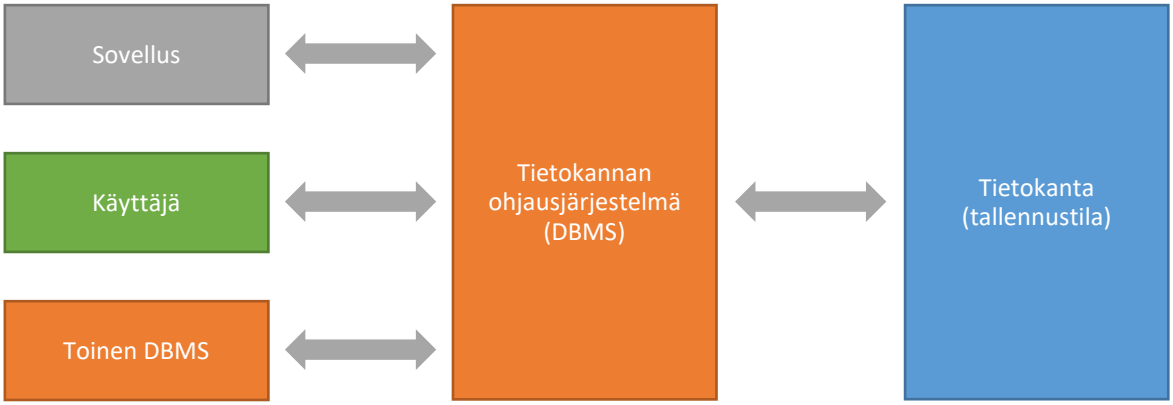
1. Laatija		Matti Meikäläinen
2. Päiväys		1.1.2007
3. Prosessi		Sisäiset tukiprosessit
4. Nimi		Kokoustilan varaaminen
5. Suorittajat		Asiakaspalveluhenkilöstö
6. Esitiedot		Kokoustilan käyttökalenteri on ajantasalla
7. Kuvaus	1	Käyttäjä aukaisee yrityksen intranetin.
	2	Käyttäjä klikkaa linkkiä ja aukaisee kokoustilavaraukset.
	3	Käyttäjä pyytää järjestelmää näyttämään kokoustilojen varaukset haluamaltaan ajalta syöttämällä viikon numero ja päivän.
	4	Käyttäjä saa eteensä varausnäytön, josta näkee kaikkien kokoustilojen vapaat ajat ja varauksen tekijöiden nimet.
	5	Käyttäjä valitsee sopivan kokoustilan ja varaa sen.
	6	Käyttäjä saa tilan varauksesta vahvistuksen omaan sähköpostiinsa ja merkinnän sähköiseen kalenteriinsa
8. Poikkeukset	1	Käyttäjä valitsee vahingossa jo varatun tilan - järjestelmä ilmoittaa että kokoustila on on varattu.
9. Lopputulos		Kokoustila on varattu.
10. Muut vaatimukset	1	Järjestelmän vastausajan oltava max 2 sekuntia.
	2	Kokoustilan varausnäytön päivitys saa kestää 4 sekuntia.
	3	
	4	

Taulukko 2. Esimerkki käyttötapauslomakkeesta (JHS 173, 27)

Käyttöympäristöllä tarkoitetaan niitä laitteita tai ympäristöjä, missä ohjelmisto toimii. Käyttöympäristöön asetetut vaatimukset kuuluvat järjestelmän teknisiin reunaehtoihin. Niitä voi olla esimerkiksi laitteiston suorituskykyvaatimukset sekä käytettävät käyttöjärjestelmät. (JHS 173, 24.)

3.3 Tietokantamalli

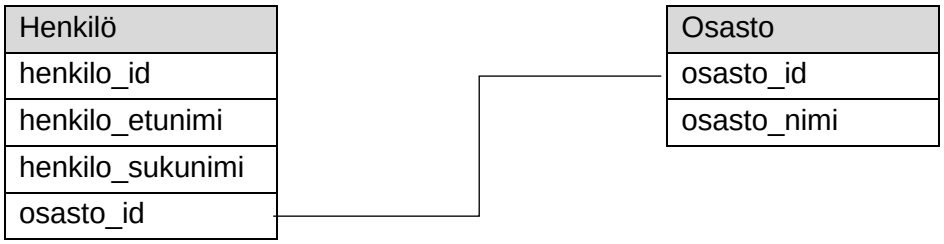
Tietokantojen tarkoitus on tietojen säilyttäminen sekä niiden käsittely mielekkäällä ja tehokkaalla tavalla. Tietokantasovellus koostuu aina kolmesta päätasosta: tietokannasta, tietokannan ohjausjärjestelmästä (DBMS) ja käyttäjästä (Kuvio 5). Käyttäjällä voidaan tarkoittaa myös sovellusta tai toista tietokannan ohjausjärjestelmää (Sharma 2019). Tietokanta tarkoittaa tiedon fyysistä tallennustilaa, mistä tietokannan ohjausjärjestelmä kääntää käyttäjille ns. loogisen tietokannan. Käyttäjä ei siis suoraan käsittele fyysisiä tietoja vaan ainoastaan siihen viittaavia osoitteita. Tämä mahdollistaa samalle tiedolle monta käyttäjää kerrallaan ja vähentää saman tiedon tallentumista palvelimille, mikä taas tehostaa tallennustilan käyttöä. (Date 2013.)



Kuvio 5. Tietokantasovelluksen arkkitehtuuri (Sharma 2019)

Tietokannan ohjausjärjestelmän voi jakaa arkkitehtuurinsa ja käyttötarkoituksensa mukaan muun muassa hierarkkiseen, verkkomalliseen, relaatiomalliseen tai oliomalliseen. Tietokantojen historian alkuvaiheessa käytössä oli lähinnä hierarkkisia, mutta tiedon lisääntyminen, käyttäjämäärien kasvaminen ja tiedostojen käyttö pakottivat kehittämään joustavampia ja tehokkaampia järjestelmiä. (Sharma 2019.)

Relaatiotietokannassa pyritään tilanteeseen, jossa saman tiedon tallentuminen useaan kertaan pyritään poistamaan. Tämä toteutetaan niin, että tiedot tallennetaan tauluihin, joista luodaan yhteyksiä toisiin tietoja sisältäviin tauluihin (Chand 2021). Jos esimerkiksi tarkoituksena on tallentaa henkilöiden nimitietoja ja heidän osastonsa organisaatiossa, ei kannata usealle henkilölle lisätä saman osaston nimeä vaan osastolle luodaan oma taulunsa, mistä luodaan viittaus henkilö -tauluun. Tämä mahdollistaa myös sen, että jos osaston nimi halutaan muuttaa esimerkiksi toiselle kielelle, riittää, että se muutetaan vain osasto -tauluun eikä henkilö -taulun jokaiselle riville erikseen (Kuvio 6).



Henkilö		
ID	**Nimi**	**Osasto**
1	*Liisa Heikkilä*	*1*
2	*Antti Mattila*	*2*
3	*Tero Taulu*	*1*

Osasto	
ID	**Nimi**
1	*Myynti*
2	*Hallinto*

Kuvio 6. Tietojen järjestäminen relaatiomallissa

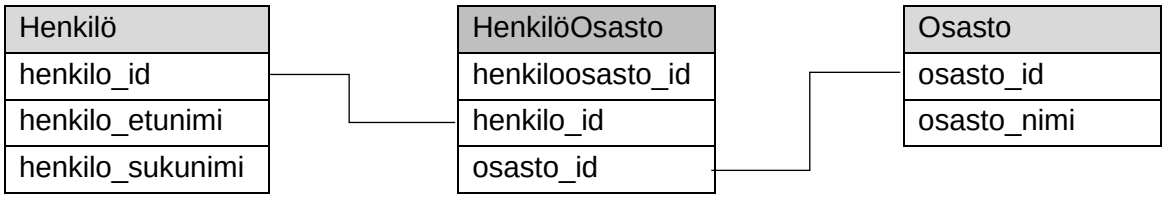
Tietokanta sisältää usein monimutkaisempia kokonaisuuksia, joten saman tiedon toistumisen minimointiin käytetään normalisointia. Normalisoinnissa on määritelty tasoja ja nykyään tasot on jaettu 7 eri normaalimuotoon (eng. *normal form*): 1NF, 2NF, 3NF, BCNF, 4NF, 5NF ja 6NF. Relaatiotietokannoissa kolmas normaalimuoto (3NF) on usein riittävä tavoite, sillä siitä korkeammat normaalimuodot tulevat tarpeeseen vain, jos tietokanta on erittäin monimutkainen ja laaja. (Peterson 2021.)

Normalisointi tapahtuu asteittain ensimmäisestä muodosta ylöspäin. Tietokanta ei siis voi olla toisessa normaalimuodossaan (2NF), jos ensimmäisen normaalimuodon (1NF) kriteerit eivät täyty (Taulukko 3).

Aste	**Kriteerit**
1NF	• Jokainen solu sisältää vain yhden arvon • Jokainen rivi on ainutlaatuinen
2NF	• 1NF toteutuu • Jokainen avaimeen kuulumaton attribuutti on täydellisesti riippuvainen avaimesta
3NF	• 2NF toteutuu • Mikään avaimeen kuulumattomista attribuuteista eivät ole keskenään riippuvaisia

Taulukko 3. Normaalimuotojen kriteerit (Peterson 2021)

Taulukossa 3 esiintyvällä attribuutilla tarkoitetaan relaatiotauluun kuuluvia ominaisuuksia. Esimerkiksi Opiskelija-taulun attribuutteja voisivat olla opiskelijanumero, etunimi ja sukunimi. Jotta tietokanta saadaan normalisoitua kolmanteen normaalimuotoon, taulujen väleiltä on poistettava kaikki monesta-moneen-viittaukset. Edellistä esimerkkiä käyttäen: jos henkilön on mahdollista toimia useammassa osastossa samanaikaisesti, taulujen väliin on muodostettava välitaulu (Kuvio 7).



Kuvio 7. Välitaulun luominen päätaulujen väliin

3.4 Käyttöliittymä

Käyttöliittymällä (eng. *UI - User Interface*) tarkoitetaan sitä osaa ohjelmistoa, tietokonetta tai verkkosivua, mitä kautta ihminen käyttää niihin liittyviä toimintoja (Indeed 2021). Käyttöliittymän suunnittelussa on otettava huomioon mm. visuaalisuus, vuorovaikutus ja tietojen saavutettavuus (Murphy 2019).

Käyttöliittymästä, ja tuotteesta yleisestikin, syntyy käyttäjälle erilaisia tuntemuksia, joita kutsutaan käyttökokemuksiksi (eng. *UX - User Experience*). Vaikka käyttöliittymä on enemmän tekninen toteutus ja käyttökokemusten arviointi lähempänä psykologiaa, niiden huomioiminen käyttöliittymän suunnittelussa menevät usein limittäin. Eroavaisuudet voidaan ymmärtää parhaiten tarkastelemalla, mitä asioita käyttöliittymän suunnittelussa ja käyttäjäkokemuksen suunnittelussa pyritään huomioimaan (Taulukko 4). (Masters In Data Science 2020.)

Käyttöliittymä	**Käyttäjäkokemus**
Väriteema	Hyödyllisyys
Typografia	Käytettävyys
Ulkoasu	Löydettävyys
Kuvat	Kätevyys
Lomakkeet	Uskottavuus
Painikkeet	Saavutettavuus
Sisältö	Haluttavuus

Taulukko 4. Käyttöliittymän ja käyttäjäkokemuksen suunnittelussa huomioitavat asiat (SPD Load 2019)

Käyttöliittymä on siis yksi keskeinen osa käyttäjäkokemuksen muodostumista ja taulukossa 4 esitetyissä käyttöliittymän suunnittelun osa-alueilla pyritään pääsemään mahdollisimman monessa käyttäjäkokemuksen osa-alueessa hyvään tulokseen. Se, että osa käyttäjäkokemuksen kriteereistä voi olla keskenään ristiriitaisia, vaikeuttaa käyttöliittymän suunnittelua entisestään: suuriresoluutioisilla kuvilla voidaan herättää käyttäjän mielenkiinto, mutta usein niiden takia joudutaan lisäämään näytön vierittelyä, mikä voi vähentää löydettävyyden tai käytettävyyden tuntemuksia. (Masters In Data Science 2020.)

Käyttöliittymän suunnittelussa tulee huomioida myös käyttöympäristön antamat vaatimukset. Jos sovellusta on mahdollista käyttää tietokoneella ja mobiililaitteilla, olisi ulkoasun oltava responsiivinen eli skaalautuva eri kokoisille näytöille (Friedman 2018). Vuonna 2020 kerätyt verkkosivujen vierailutiedot antavat osviittaa siitä, että varsinkin selainpohjaisissa toteutuksissa tulee erittäin vahvasti huomioida responsiivisuus: maailmanlaajuisesti 68,1 % sivustovierailuista tehtiin mobiililaitteilla, mutta 53,3 % sivustoilla vietetystä ajasta tapahtui silti tietokoneilla (Enge 2021).

Väreillä voidaan käyttää hyväkseen ohjaamaan tai avustamaan käyttäjän valintoja. Vaikka samalla värillä voi olla eri kulttuurissa täysin erilainen tarkoitus, laajojen psykologisten tutkimusten perusteella on havaittu muutamia yhteneväisyyksiä. Länsimaissa punainen yhdistetään vaaraan tai virheeseen, kun taas vihreä yhdistetään turvallisuuteen ja edistymiseen (Eriksen 2020). Tämän lisäksi ohjelmistoissa on jo vuosikymmeniä yleistynyt tietyt värikäytänteet, joten kuka tahansa kaupallisia ohjelmistoja käyttänyt on altistunut tietyille johdonmukaisuuksille: ohjelman sulkemiseen liittyvä nappi on lähestulkoon aina punainen, mikä varoittaa, että esimerkiksi tallentamattomat tiedot saattavat kadota (Babish 2019).

Käyttöliittymän värimaailman valitseminen on usein haastavaa, sillä vaihtoehtoja on loputtomasti, mutta silti useasti valitut värit eivät sovi yhteen keskenään. Se, että käytetyt värit saadaan toimimaan miellyttävästi keskenään, kutsutaan värien harmoniaksi (Hannah 2021). Värimaailman suunnittelua varten ei tarvitse erikoistua värien teoriaan perustasoa syvemmälle, sillä värien valitsemiseen on olemassa useita ilmaisia työkaluja kuten Adobe Color (https://color.adobe.com/create/color-wheel) tai Coolors (https://coolors.co/).

4 Sovelluksen suunnittelu ja toteutus

4.1 Sovelluksen suunnittelu

Sovelluksen suunnittelu aloitettiin aloituspalaverilla, jossa koulutuksen suunnitteluun osallistuville henkilöille (pääkäyttäjät) esiteltiin nykytilanteen puutteet ja ongelmat sekä esiteltiin, miten nykyisen järjestelmän korvaaminen tietokantapohjaisella sovelluksella vaikuttaisi suunnittelutyön toteuttamiseen ja raportointiin. Koska pääkäyttäjillä ei kellään ollut yhtään kokemusta sovelluskehityksestä, palaverissa pyrittiin esimerkein luomaan realistista kuvaa, minkälaisia ominaisuuksia sovellukseen olisi tällä aikataululla mahdollista toteuttaa.

Projekti suunniteltiin toteutettavaksi 6 vaiheessa: alkukartoitus, vaatimusmäärittely, tietokantasuunnittelu, käyttöliittymän suunnittelu, sovelluksen toteuttaminen sekä testaus ja raportointi.

Alkukartoitusvaiheessa toteutettiin myös tutkimuslupahakemus, mikä vaaditaan kaikissa Puolustusvoimille toteutettavissa tutkimuksissa ja lopputöissä sekä selvitettiin sovelluskehityksessä esiintyvät varmat reunaehdot. Tutkimuslupahakemukseen hahmoteltiin projektille alustava aikataulu (Taulukko 5).

Vaihe	**Alkaa**	**Päättyy**	**Kesto**
Vaatimusmäärittely	27.09.2021	29.09.2021	3 pv
Tietokantasuunnittelu	30.09.2021	04.10.2021	5 pv
Käyttöliittymän suunnittelu	05.10.2021	07.10.2021	3 pv
Sovelluksen toteuttaminen	08.10.2021	12.11.2021	35 pv
Testaus ja raportointi	15.11.2021	26.11.2021	12 pv

Taulukko 5. Projektin alustava aikataulu

Vaiheiden aikana sovittiin toteutettavaksi erikseen sovittava määrä tarkasteluja ketterän kehitystavan mukaisesti. Vaiheesta riippuen käsiteltävien asioiden määrä ja aiheet vaihtelivat. Tarkastelut toteutettiin työyksikön normaalien palaverien yhteyksissä, sillä palavereissa oli jo muutenkin budjetoitu aikaa viikko- ja tuntisuunnitelmien esittelyyn sekä niihin liittyvän hallinnon tarkasteluun ja kehittämiseen. Aikatauluun ei myöskään suunniteltu erillistä vaihetta opinnäytetyöraportin kirjoittamiseen vaan raporttia täydennettiin projektin aikana ja se oli osa projektin dokumentointia. Kuitenkin painopiste raportin kirjoittamiseen ajoittui sovelluksen suunnittelu- ja raportointivaiheeseen.

Vaatimusmäärittely

Ohjelmiston suunnittelu aloitettiin vaatimusmäärittelyllä. Työn vaatimusmäärittely toteutettiin kahdessa vaiheessa: ensin tehtiin hahmotelman sovelluksen sisällöstä ja toiminnoista, mikä esiteltiin pääkäyttäjille, minkä jälkeen pääkäyttäjiltä kysyttiin ensimmäiset havainnot ja mielipiteet hahmotelmasta. Niistä syntyvät ideat käsiteltiin välittömästi: toteutettavat ja tarkoituksenmukaiset vaatimukset kirjattiin vaatimusluetteloon ja aikataulun kannalta mahdottomat vaatimukset jätettiin projektin ulkopuolelle. Tämän lisäksi vaatimusluettelo jaettiin kaikkien käyttöön ja ohjeistettiin kirjaamaan omia huomioita sitä mukaan, kun niitä ilmenee. Koska pääkäyttäjillä ei ollut hirveästi kokemusta sovelluskehityksestä, suurin osa laadukkaista vaatimuksista saatiinkin juuri myöhemmin ja vähitellen.

Kun vaatimusluettelo oli saatu pääosin valmiiksi, niissä esitetyt vaatimukset priorisoitiin ja luokiteltiin niin, että niistä oli selkeästi havaittavissa osakokonaisuuksia. Koska tietokantasovelluksen ydin ei ole itse sovellus vaan siihen syötetty data, sovellus päätettiin rakentaa pieni pala kerrallaan, jotta tiedon syöttäminen voitaisiin aloittaa mahdollisimman aikaisessa vaiheessa. Vaikkei pääkäyttäjillä ollutkaan riittävää osaamista puuttua itse sovelluksen kehittämiseen, tavoitteena oli kuitenkin edetä ketterän kehitystavan mukaisesti ja sovelluksen käyttöönotto vaiheittain kätteli hyvin tämän menetelmän kanssa. Aina, kun yksi osakokonaisuus saatiin tuotettua, se esiteltiin käyttäjille ja ohjeistettiin, miten sitä sovellusosaa voisi alkaa jo käyttämään. Lisäperusteluna tälle oli myös projektin tiukka aikataulu eli vaikka sovelluksen käyttöönotto vaiheittain oli riski mm. myöhäisten muutosten vaikeuden takia, olisi samat riskit olleet olemassa myös projektin lopussa käyttöönotettavan valmiin tuotteen kanssa.

Projektin edetessä huomattiin, että kaikkia vaatimuksia ei tulla saamaan projektille asetetussa aikataulussa toteutettua. Tämän johdosta päätimme lisätä projektiin myös jatkokehitysvaiheen, mihin siirsimme osan projektiin hyväksytyistä vaatimuksista sekä nostimme sinne mukaan myös vaatimusmäärittelystä ajan takia ulosrajattuja vaatimuksia. Jatkokehitysvaiheeseen siirtyvät lähinnä sovelluksen toimintaa helpottavia vaatimuksia sekä laajemmat raportointimahdollisuudet. Ne eivät olleet sovelluksen toiminnan kannalta kriittisiä ja näin ollen ne voitiin toteuttaa vasta, kun sovellus on jo otettu käyttöön. Tässä vaiheessa uudeksi vaatimukseksi nousi myös versionhallinta.

Käyttöympäristö

Puolustusvoimilla on käytössään tietoturvaluokitukseltaan monia erilaisia järjestelmiä ja ohjelmistoja. Opinnäytetyön tuloksena syntyvää sovellusta käytettäisiin ainoastaan ei-julkisissa järjestelmissä, jolloin käyttöympäristö loi selkeästi suurimman rajoituksen ohjelmiston toteuttamiselle. Tällä aikataululla ei ollut mahdollisuutta luoda Puolustusvoimien sisäiseen verkkoon tietokantasovellukseen vaadittavia verkkopalvelimia tietokantatuella, mikä olisi mahdollistanut sovelluksen toteuttamisen selainpohjaisena. Puolustusvoimilla on laaja Microsoft Office -tuoteperheen lisenssi, joten ainoaksi mahdollisuudeksi muodostui MS Access tietokantojen käsittelyohjelma. Accessissa on oma sisäinen tietokantamoottori eli se ei tarvinnut verkkolevyille erillisten ohjelmistojen asentamista.

Työn sovellus ei siis luonut käyttöympäristölle vaatimuksia vaan käytössä oleva infrastruktuuri asetti sovelluskehitykselle selkeän vaatimuksen. Vaikka valinta jouduttiin tekemään pakon sanelema, aiemmin samojen toimintojen toteuttamiseen käytettiin jo saman tuoteperheen tuotetta (MS Excel), joten vanhojen tuotteiden tallentaminen uuteen tietokantaan tapahtuisi melko vaivattomasti.

Tietokantamalli

Koulutuksen suunnittelutyökalu rakentuu yksittäisten harjoitusten tietojen ympärille: ensin rakennetaan harjoitus, mihin sitten sidotaan harjoituksen pitäjät, harjoituksen sijainnit sekä harjoituksen ajankohta. Vaatimusmäärittelyn pohjalta identifioitiin tarvittaviksi tiedoiksi:

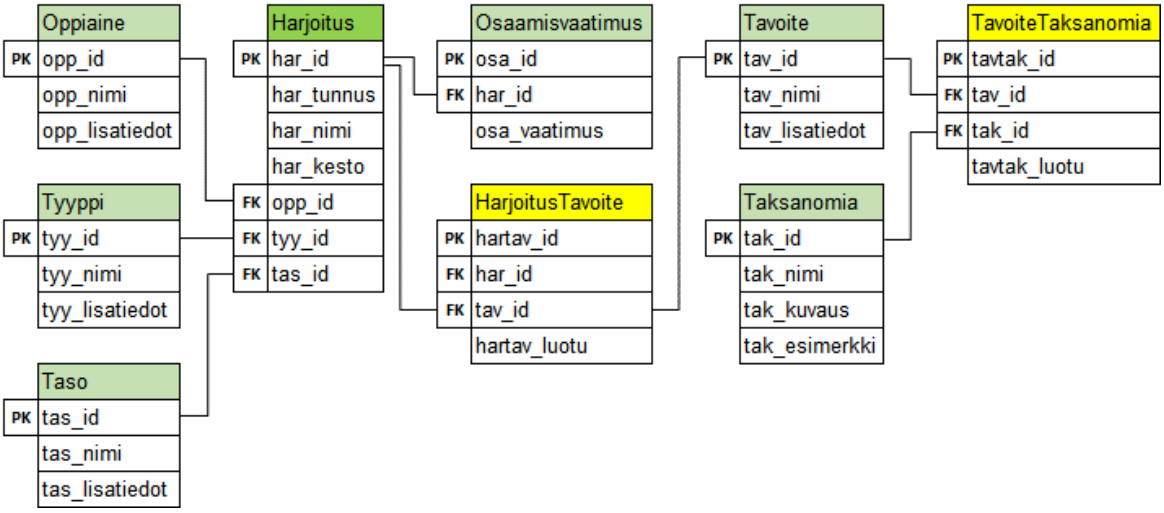
- Harjoituksen tunnus
- Harjoituksen otsikko
- Tyyppi (oppitunti, käytännön harjoite)
- Oppiaine
- Kesto
- Taso
- Hakusanat
- Lähtötasovaatimukset
- Harjoituksen sisältö ja aikautus
- Harjoituksen tavoitteet

Näiden lisäksi harjoitukselle luotiin laskurityyppinen avain (harjoitus-ID), sillä samasta harjoituksesta olisi mahdollisuus tehdä useampi versio, jolloin harjoituksen tunnus ei olisi riittävän yksilöivä attribuutti avaimeksi. Tämän jälkeen taululle toteutettiin normalisointi asteittain. Ensin arviointiin kunkin attribuutin osalta, että tulisiko se erottaa omaksi taulukseen

sen perusteella, että sisältäisikö se useampia kuin yhden arvon tai esiintyykö siinä toistuvasti samoja arvoja. Havaittiin, että tyyppi, oppiaine, taso, hakusanat, lähtötasovaatimukset ja tavoitteet olivat sellaisia attribuutteja, jotka tuli erottaa omiksi tauluiksi. Tämän jälkeen tarkastettiin, pitikö Harjoitus-taulun ja siitä erotettujen taulujen välille luoda myös välitauluja. Tarkastus toteutettiin tyypillisellä, kaksiosaisella testillä:

1. *Voiko yhdellä X:llä olla useampi Y?*
2. *Voiko yksi Y sisältää useamman X:än?*

Jos vastaus on molempiin ”kyllä”, taulujen luotiin välitaulu. Näiden testausten perusteella harjoitukseen liittyvien tietojen tietokantamalli oli kuvion 8 mukainen.



Kuvio 8. Harjoitukseen liittyvien tietojen kantamalli normalisoinnin jälkeen

Näiden tietojen välille joutui siis muodostamaan kaksi välitaulua, sillä monessa harjoituksessa voi esiintyä sama tavoite ja samalle tavoitteelle voidaan harjoituksesta riippuen luokitella eri taksanomia (tuntee, osaa, hallitsee).

Samalla prosessilla toteutettiin henkilöihin ja sijainteihin liittyvät taulut, minkä jälkeen nämä kokonaisuudet sidottiin toisiinsa. Tietokantamallin lopputuloksena syntyi taulukkolaskentatiedosto, missä esiintyy kaikki sovelluksen pää-, ala- ja välitaulut tietotyyppeineen. Sovelluksen tietokantasuunnittelusta valmistunut tietokantamalli on esitelty liitteessä 2.

Jo ensimmäisen ohjauspalaverin yhteydessä todettiin, että harjoituksen tekijä olisi hyvä lisätä harjoituksen attribuutteihin sitä varten, että jos itse harjoituksesta ilmenisi jotain kysyttävää, niin kysymykset osattaisiin esittää oikealle henkilölle. Koska yhdellä harjoituksella on vain yksi luoja, oli Harjoitus-taulun loppuun helppo tehdä viittaus Henkilö-tauluun. Palaverin

päätteeksi informoitiin osapuolia siitä, miten tietokantamallin muutokset voisivat vaikuttaa projektin aikatauluun ja toteuttamiseen.

Käyttöliittymä

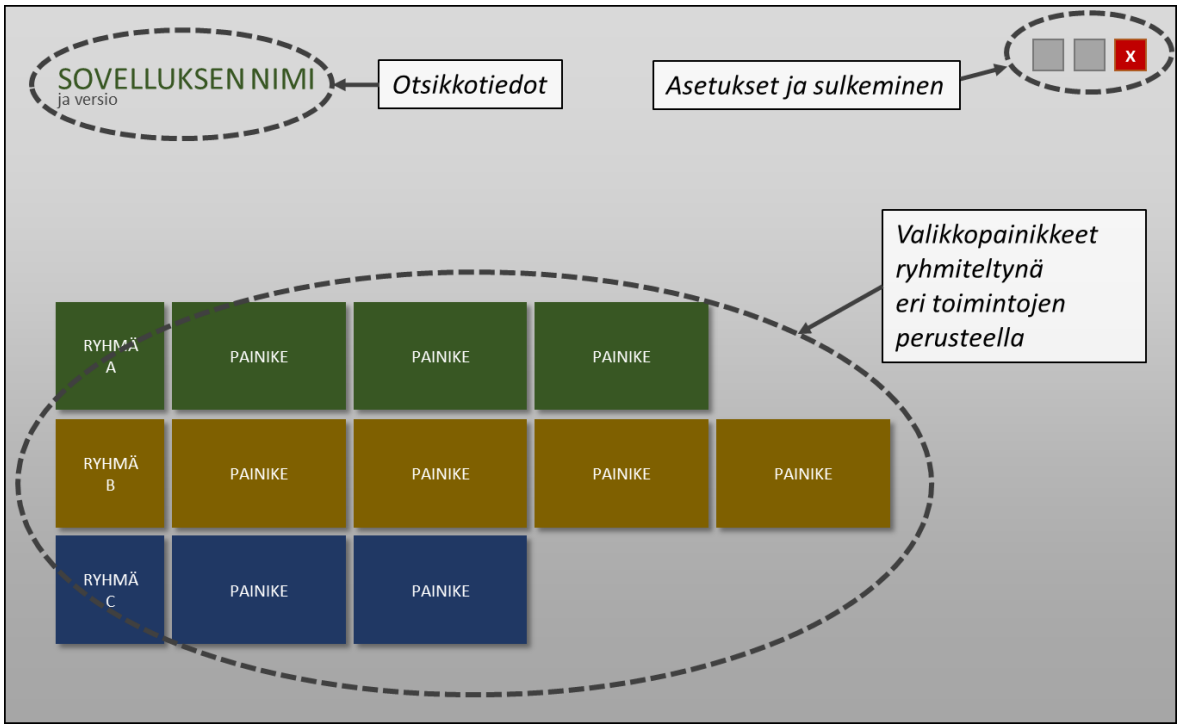
Käyttöliittymän suunnittelussa tulee ottaa huomioon käyttäjäkunta ja ohjelmiston tarkoitus. Tässä tapauksessa ohjelmistoa ei yritetä sanan varsinaisessa merkityksessä myydä mahdollisimman usealle käyttäjälle, mutta käyttökokemuksella on suuri vaikutus siihen, kuinka tehokkaasti käyttäjät saadaan sitoutettua sovelluksen käyttöön ja kuinka tehokkaasti he voivat käyttää heille varattua suunnitteluaikaa. Yksi sovelluksen tärkeimpiä tavoitteita kuitenkin oli helpottaa suunnittelutyötä ja vapauttaa henkilöresursseja käytännön koulutustehtäviin. Tämän takia sovelluksen tulisi olla käytettävyydeltään mahdollisimman vaivaton sekä visuaalisuudeltaan omaksuttava.

Käyttäjäkunnan huomioiminen myös osaamisen kautta on erittäin tärkeää. Tulevat käyttäjät ovat tottuneita yleisimpien toimistotyökalujen (taulukkolaskenta, tekstinkäsittely, diaesitys) käyttöön, joten sovelluksen arkkitehtuurissa kannattaisi jäljitellä näiden ohjelmien toimintoja sekä ulkoasua. Eritoten taulukkolaskentatiedostoja on jokainen käyttäjä käyttänyt jo vuosia, jolloin tietojen käsittelyssä kannattaakin suosia mieluummin taulukkomuotoja kuin lomakemuotoja.

Sovelluksen käyttöliittymä muodostuu käytännössä kolmesta pääelementistä: päävalikko, lomakkeet ja raportit. Päävalikon kautta käyttäjä pääsee suoraan kiinni haluamaansa toimintoon, lomakkeiden kautta lisätään, muokataan tai poistetaan tietoja ja raporttien avulla tulostetaan tarvittavat tiedot.

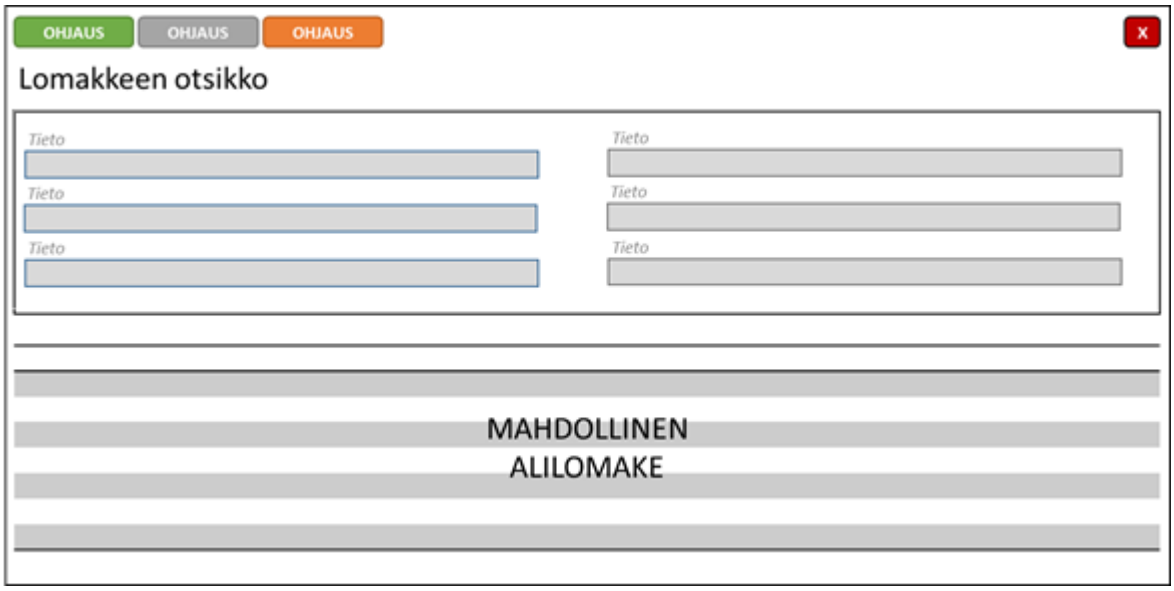
Päävalikon suunnittelu alkoi siitä, että sovelluksesta pyrittiin identifioimaan kaikki halutut lomakkeet ja raportit, joista pyrittiin koostaa toiminnoiltaan samankaltaisia ryhmiä. Tämä auttaisi hahmottamaan, kuinka monia eri painikkeita sovelluksen alkuvalikkoon pitäisi saada sijoitettua. Koostamisen pohjalta ryhmiksi muodostui kolme kokonaisuutta: luo, rakenna ja tulosta. Luo -ryhmä tarkoittaa harjoitusten ja siihen liittyvien attribuuttien, kuten oppiaineiden, tyyppien ja henkilöiden, luomista ja muokkaamista. Rakenna -ryhmässä luotuihin harjoituksiin sidottaisiin tapahtuma-aika, sijainti ja henkilöstö, minkä jälkeen "tulosta" -ryhmässä voidaan tulostaa mm. tuntisuunnitelmat, viikko-ohjelmat sekä yksittäisten harjoitusten sisällöt ja tavoitteet.

Tämän jälkeen päävalikon rakenne hahmoteltiin Powerpoint -työkalulla kuvion 9 kaltaiseksi. Moni suunnittelija suosii ensimmäisten hahmotelmien tekemistä pelkälle paperille mm. tarralappujen avulla. Moni taas kokee sähköisten työkalujen käytön helpompana, sillä elementtien kokojen muuttaminen ja uudelleensijoittelu on vaivattomampaa.



Kuvio 9. Päävalikon konsepti

Sovellukseen tulisi useita lomakkeita ja niissä esiintyvien tietojen sommittelu vaihtelee huomattavasti riippuen siitä, mitä tietoja lomakkeella käsitellään. Harjoituksen luomiseen liittyvä lomake sisältää useita kymmeniä tietoja, kun taas henkilön lisäämiseen luotu lomake sisältää vain muutaman erillisen tiedon. Vaikka lomakkeen koko ja syötettävien tietojen määrä vaihtelevat, ulkoasun suunnittelussa on kyettävä tunnistamaan lomakkeilla ilmenevät samankaltaisuudet ja niiden kohdalla pyrkiä johdonmukaisuuteen. Näitä voi olla esimerkiksi tietojen muokkaamiseen liittyvät painikkeet, esitettävien tietojen sijainti sekä värit (Kuvio 10).



Kuvio 10. Tiedonsyöttölomakkeiden konsepti

4.2 Sovelluksen rakentaminen

4.2.1 Taulut

Sovelluksen rakentaminen aloitettiin tietokannan luomisella eli tarkemmin sanottuna taulujen luomisella. Microsoft Access -työkalu käyttää tietokantojen käsittelemiseen SQL-kieltä ja kaikki tietokantatoiminnot on mahdollista toteuttaa suoraan koodikielellä. Ohjelmistoon on kuitenkin helpotukseksi toteutettu myös ohjattuja toimintoja ja visuaalinen käyttöliittymä yleisimpien toimien toteuttamiseksi. Esimerkiksi taulujen luominen on kätevää suoraan rakennenäkymässä. Rakennenäkymässä kullekin attribuutille luodaan yksilöivä nimi sekä tietotyyppi (Kuva 1). Näkymässä on mahdollisuus vaikuttaa myös tarkempiin asetuksiin eli syöttörajoitteisiin tai kelpoisuussääntöihin. Toisin sanottuna kaikki, mitä puhtaalla SQL-komennolla voitaisiin toteuttaa, voidaan toteuttaa myös rakennenäkymässä. Molemmissa tapauksissa on oltava hyvä tietämys relaatiotietokantojen toiminnasta ja taulujen luomisista siihen liittyen.

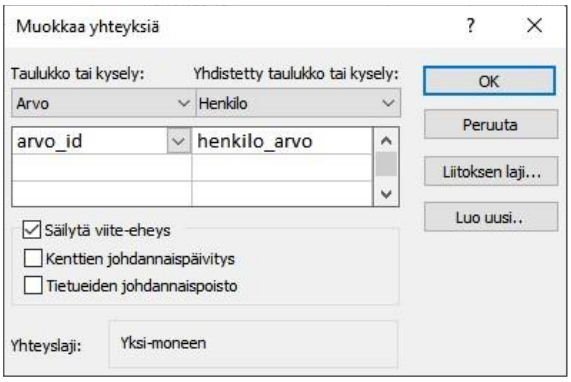
Kuva 1. Taulujen luominen rakennenäkymässä

Jokaisesta taulusta on löydyttävä yksilöivä avain, jotta relaatiomalli saadaan toimimaan. Kuvan 1 Henkilo-tauluun on luotu henkilo_id -attribuutti tätä varten. Henkilön SAP-id voisi myös toimia yksilöivänä tunnuksena, mutta koska organisaatiossa on jatkuvasti mahdollisuus toiminnanohjausjärjestelmän vaihtumiseen, ei voitu olla täysin varma, että kyseinen ID olisi käytössä esimerkiksi vielä 10 vuoden kuluttua. Tietotyyppi ”laskuri” tarkoittaa, että kullekin riville luodaan lukujonosta seuraava numero ja näin varmistetaan, ettei millään rivillä ole samaa tunnusta. Laskuri toimii myös niin, että jos henkilö, jolle luodaan tunnukseksi numero 10 ja tämä henkilö poistetaan, niin seuraavan henkilön tunnus on silti numero 11. Tämä mahdollistaa sen, että jos henkilö on historian saatossa sidottu joihinkin harjoituksiin, ei tämä henkilö korvaannu toisella henkilöllä myöhemmin.

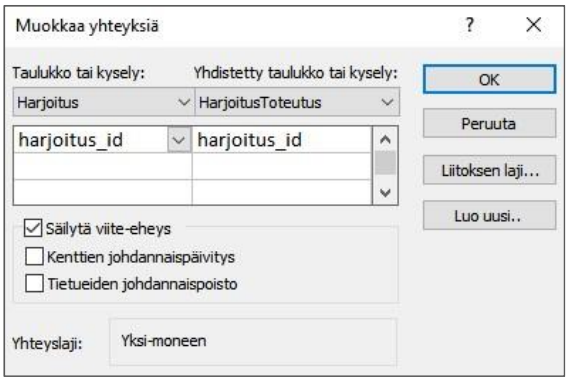
Sovellukseen luotiin 14 päätaulua, 7 välitaulua sekä 2 sovelluksen toimintaan liittymätöntä taulua (versiohallinta ja kehitysehdotukset). Taulujen yhdistämistä varten piti luoda tauluihin viiteavaimia (eng. *foreign key*). Henkilo-taulussa esiintyy yksi viiteavain, joka on henkilo_arvo. Tämän viiteavaimen tietotyyppi on luku, koska sillä viitataan Arvo-taulun arvo_id -attribuuttiin.

Sovelluksen nimeämiskäytäntö on kaikissa attribuuteissa mallin sama: taulukon nimi, alaviiva ja kentän sisältöä kuvaava nimi. Tämä helpottaa sovelluksen kehittämistä myöhemmin, kun tavoitteena on luoda monta taulukkoa yhdistäviä lomakkeita. Ilman nimeämiskäytäntöä esimerkiksi sijainnin ja harjoituksen yhdistämisessä olisi kaksi attribuuttia, joista molempien nimi olisi ”nimi” eli sekaantumisriski näiden välillä olisi suuri. Kun attribuuttien nimet ovat ”sijainti_nimi” ja ”harjoitus_nimi”, ei sekaantumisriskiä ole. Taulujen välisten yhteyksien erotteluun on myös käytetty erilaista nimeämistä: jos taulu on monesta-moneen -suhteen takia luotu välitaulu, yhteyden molemmissa päissä on identtinen nimeäminen. Toisin sanottuna yhteys on viiteavaimesta toiseen viiteavaimeen. Jos yhteys on suora viittaus yhdestä päätaulusta toiseen päätauluun, eli pääavaimesta viiteavaimeen, viiteavaimen nimeämisessä käytetään päätaululle ominaista nimeämiskäytäntöä (Kuva 2).

Pääavain - viiteavain



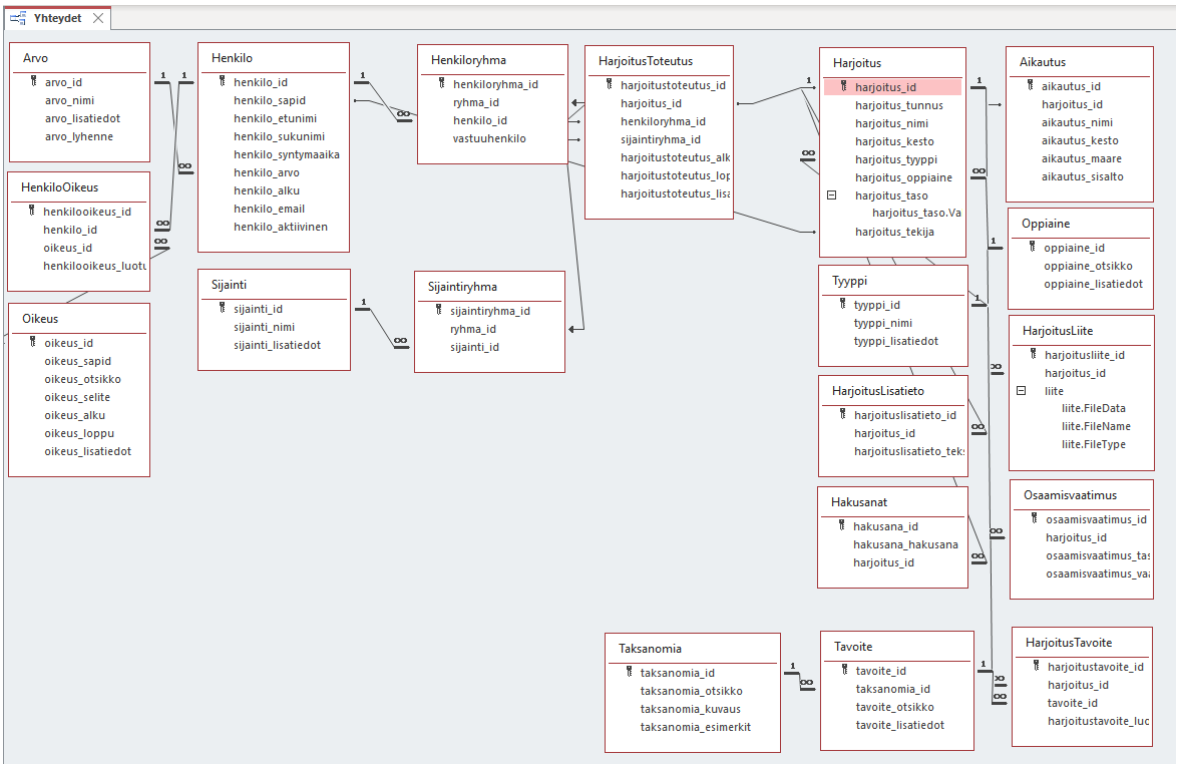
Viiteavain - viiteavain



Kuva 2. Nimeämiskäytäntö taulujen välisissä yhteyksissä

Yhteyksien välille voidaan määrittää myös yhteyssääntöjä: viite-eheys, kenttien johdannaispäivitys sekä tietueiden johdannaispoisto. Tietokantaohjelmassa esiintyvä termi kenttä tarkoittaa samaa kuin attribuutti. Viite-eheyden ollessa päällä viitatun avaimen kenttään ei voida syöttää sellaista arvoa, mitä ei ole luotu päätaulun avainkenttään. Samoin sellaisten tietueiden poistaminen päätaulusta, joista on viittaus toiseen tauluun, on mahdotonta. Tietueita voidaan kuitenkin muokata tai poistaa ilman viite-eheyden rikkomista valitsemalla sekä kenttien johdannaispäivityksen että tietueiden johdannaispoiston. Näin ollen, jos päätaulusta poistetaan pääavaimen sisältämä tietue, myös kaikki tietueet, joissa on viittaus tähän pääavaimeen, poistuvat. Tästä voi monesti syntyä peruuttamatonta vahinkoa, jos ohjelmiston käyttöön perehtymätön alkaa poistelemaan tietueita tietämättään. Jotta näiltä tilanteilta vältyttäisiin, työssä suosittiin viite-eheyden säilyttämistä, mutta johdannaispoistot ovat vain poikkeustapauksessa käytössä. Koska tietoja tulee kantaan verrattain vähän, riittää, että kaikki käytöstä poistetut tiedot jäävät vain kantaan ”turhina” kuin että poistetaan oikeasti tärkeitä tietoja. Kenttien johdannaispäivitystä ei tarvita, jos kaikkien taulujen avaimet ovat laskurityyppisiä.

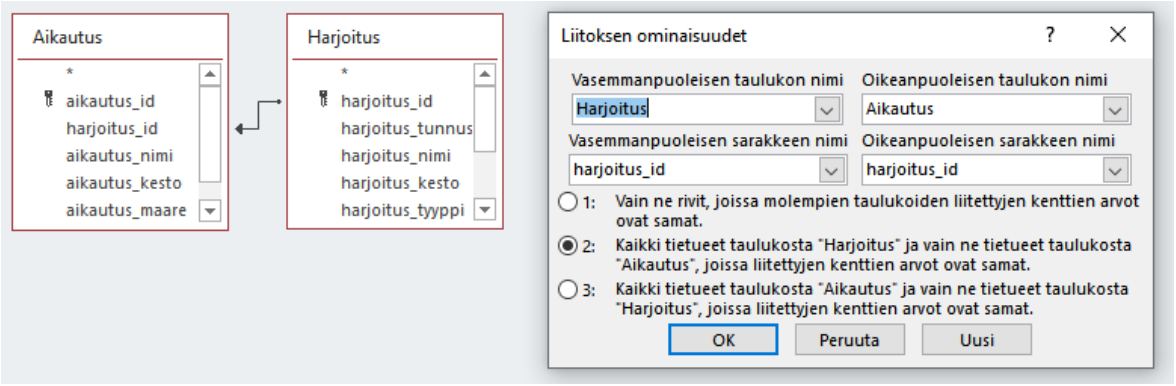
Koska ohjelmassa on yhteyksien muodostamiseen erillinen yhteydet -ikkuna, ei taulujen luontijärjestyksellä ollut merkitystä. Useissa selainpohjaisissa sovelluksissa, missä taulujen luominen toteutetaan SQL-kielellä, yhteydet lisätään monesti taulujen luontivaiheissa. Jotta käskyt saadaan toteutettua, on viiteavaimeen osoittava taulu oltava luotuna tietokantaan ennen kuin siihen on mahdollista viitata. Sovelluksen yhteydet ja Accessin yhteydet -ikkuna ovat esitelty kuvassa 3.



Kuva 3. Sovelluksen yhteydet

Taulujen ja yhteyksien muodostamisen jälkeen tietokannan toiminta testattiin syöttämällä tauluihin testidataa. Testidatan avulla simuloitiin myöhemmin lomakkeilla toteutettava tietojen syöttö ja yhteyssääntöjen toimivuus. Tiedon lisäämisessä testattiin ennen kaikkea syöttörajoitukset sekä tietotyyppien toimivuus sovelluksen kontekstissa. Tämän jälkeen tietoja yritettiin muokata ja poistaa, millä pyrittiin selvittämään mahdolliset ongelmakohdat ja heikkoudet: tärkeimpänä, pystyykö yksittäinen käyttäjä vahingossa poistamaan tarvittavia tietueita. Testaus toteutettiin niin, että sovelluksen toiminnan kannalta päätauluihin syötettiin jo mahdollisimman paljon niitä tietoja, mitä niissä todennäköisesti käytettäisiin. Taulut, kuten arvo, oppiaine ja taksanomia, sisältävät ainoastaan tällä hetkellä kaikki tiedossa olevat vaihtoehdot. Sen jälkeen niihin viittaaviin tauluihin lisättiin testidataa ja yhteysväli testattiin. Koska sovelluksessa on pääsääntöisesti kenttien johdannaispäivitys sekä tietueiden johdannaispoisto evätty, selkeitä haavoittuvuuksia ei tässä vaiheessa havaittu. Kun jokainen yksittäinen yhteysväli oli testattu, koitettiin vielä kyselyjen avulla toteuttaa harjoituksen toteutuksen tulostaminen ja näiden tulosten rajaaminen. Tarkoituksena oli siis testata koko tulevan sovelluksen ”haastavin” toiminta ja testin aikana huomattiin, että liitoksen ominaisuuksiin tulisi tehdä moneen yhteyskohtaan muutoksia. Kuvassa 4 on esitelty harjoituksen ja sen aikautuksen liitokseen tehty muutos eli harjoitus -taulusta tulisi kyselyssä ottaa kaikki tulokset huomioon ja aikautustaulusta vain sellaiset, missä sama harjoitus_id esiintyy. Ac-

cess luo oletuksena liitoksen lajiksi vaihtoehdot 1 eli tuloksissa huomioidaan vain ne tietueet, joissa ilmenee sama harjoitus_id. Tämä ei ollut sovelluksen toiminnan kanssa sopiva, sillä harjoitus oli mahdollista luoda myös ilman aikautusta. Sama ongelma ilmeni harjoituksen henkilöstön ja sijaintien kanssa.



Kuva 4. Liitoksen ominaisuudet -ikkuna

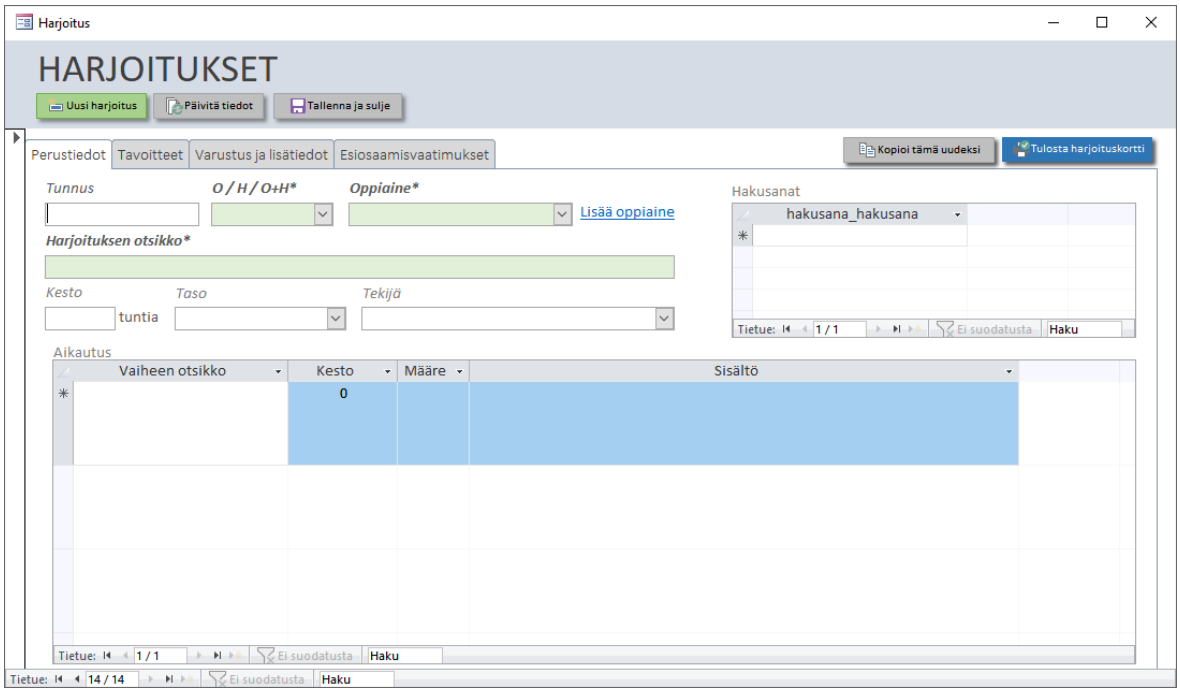
4.2.2 Lomakkeet

Kun sovellukseen liittyvät oleellisimmat tietokantatoiminnot oli testattu, alettiin rakentamaan sovelluksen lomakkeita. Selainpohjaisissa sovelluksissa lomake mielletään yleensä tietojen keräämiseen tarkoitettuna elementtinä (form), joka lisätään HTML-sivulle. Accessissa ei erikseen luoda sivuja vaan kaikki ulkoasuun, navigoimiseen ja tietojen keräämiseen tarkoitetut osiot ovat lomakkeita. Toisaalta tämä suoraviivaisti sovelluksen tietokantatoimintojen tekoa, mutta monimutkaisti huomattavasti kaikkea graafista toteutusta.

Lomakkeiden luomisessa käytettiin samaa kaavaa kuin taulukoiden luomisessa. Ensin aloitettiin ns. päätepisteistä eli luotiin lomakkeet tauluista, joihin ei ollut viittauksia ulkoapäin. Näitä olivat esimerkiksi sijainti, oikeus tai oppiaine. Tämän jälkeen luotiin askel askeleelta haastavammat kokonaisuudet eli henkilön luominen, harjoituksen luominen ja viimeisenä harjoituksen liittäminen viikko-ohjelmaan (harjoitustoteutus).

Harjoitustaulun luomisessa huomattiin, että lomakkeelle tulisi todella paljon erilaisia tietoja, joista osa oli vielä sellaisia, johon oli mahdollisuus syöttää monta arvoa. Jotta lomakkeelle saataisiin kaikki tiedot mahtumaan ilman, että käyttäjä joutuisi huomattavasti rullaamaan näkymää pysty- tai vaakasuorassa, piti lomaketta vähän suunnitella lisää. Ratkaisuksi löytyi Accessin ohjausobjekteista välilehti -objekti. Harjoituksen tietojen syöttö jaettiin ryhmiin ja kullekin ryhmälle luotiin oma välilehtensä. Välilehtien ryhmittelyssä otettiin huomioon, mitä tietoja on pakko syöttää, mitä tietoja yleensä syötettäisiin ja mitä kaikkia tietoja voitaisiin

syöttää mielekkäästi tämän lomakkeen kautta. Kuvassa 5 näkyy lopullinen Harjoitus -lomake. Lomakkeeseen kuuluu yhteensä 5 alilomaketta, joten niiden tiivistäminen yhdelle välilehdelle olisi haitannut lomakkeen käytettävyyttä huomattavasti.



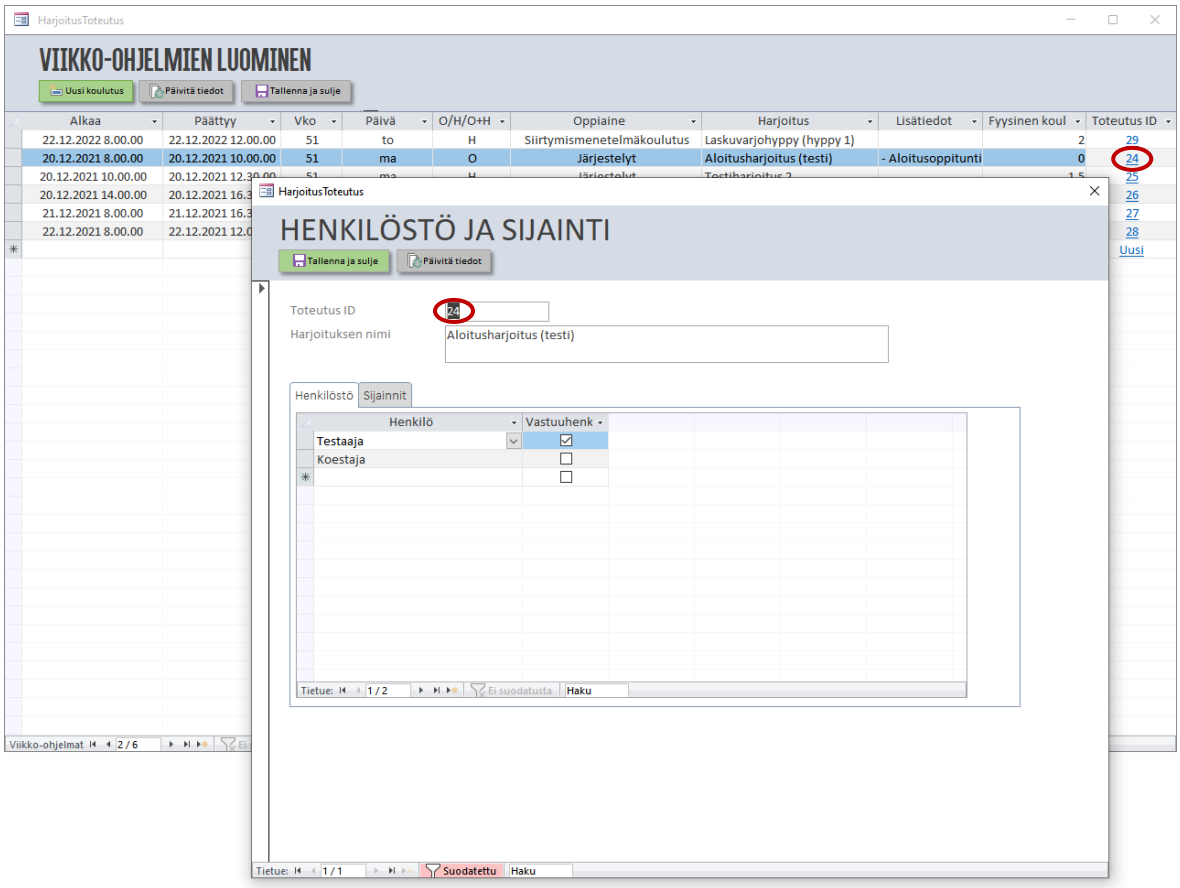
Kuva 5. Harjoitus -lomake

Alilomakkeiden muodostaminen on tietokantatyökalulla melko helppoa, kunhan lomakkeiden luomisen logiikan ymmärtää. Ohjattu lomakkeen luominen mahdollistaa usein vain yhden alilomakkeen muodostamisen, joten alilomakkeet olivat järkevämpää luoda vasta jälkikäteen. Kuvassa 5 esiintyvä lomake luotiin siis ensin pitämään sisällään tiedot tunnus, tyyppi, oppiaine, otsikko, kesto, taso ja tekijä. Kaikki tiedot löytyivät siis harjoitustaulusta. Kun niiden pohjalta oli luotu lomakepohja, lomakkeelle lisättiin alilomake -ohjausobjekti. Tämän takia taustalla olevalle päälomakkeelle piti tietokantahaussa ottaa mukaan myös harjoitus_id, vaikka sitä tietoa ei käyttäjälle näytettykään: kuten kuvassa 3 näkyy, kaikkien erillisten taulujen viittaukset ovat harjoitustaulun harjoitus_id -attribuuttiin.

Haastavin kokonaisuus oli viikko-ohjelmien luominen. Lomakkeessa pitäisi yhdistää luotuja harjoitus -olioita, harjoituksen henkilöstö, harjoituksen sijainti sekä harjoituksen ajankohta. Oliolla tarkoitetaan kunkin harjoituksen luomiseen käytettävien attribuuttien sisältämää tietojoukkoa eli jokainen olio sisältää samat attribuutit, mutta niihin tallennetut tiedot vaihtelevat. Lomakkeelle tulisi siis useita monesta-moneen-liitosten yhdistelmiä: sama harjoitus voidaan pitää useita kertoja eri henkilöiden toimesta ja eri sijainneissa, mutta toisaalta samat

henkilöt voivat pitää eri harjoituksia samassa sijainnissa. Ainut ainutlaatuisuus oli tapahtuma-aika eli sama henkilö ei voinut olla samaan aikaan useassa eri harjoituksessa mukana.

Viikko-ohjelmien luominen päätettiin toteuttaa kaksivaiheisena. Ensimmäisessä vaiheessa käyttäjälle avautuu taulukkomainen lomake, johon voi listata tapahtuma-aikoja ja näihin aikoihin luotuja harjoituksia. Tämän lomakkeen kautta voi myös lennosta lisätä harjoituksia: jos viikko-ohjelmaa luotaessa haluttua harjoitusta ei vielä ole luotu, käyttäjä siirretään harjoituslomakkeelle. Harjoituslomakkeella käyttäjä lisää harjoitukseen vähintään pakolliset tiedot, minkä jälkeen lomakkeelta poistuttaessa tämä harjoitus on valittavissa viikko-ohjelman luomisessa. Kun harjoitukselle on luotu tapahtuma-aika, tähän ainutlaatuiseen toteutukseen on mahdollista kirjata vielä lisätietoja. Jokaiselle toteutukselle luodaan avaimeksi ”Toteutus ID” ja painamalla ID:tä, avautuu ponnahdusikkuna, missä käyttäjä pääsee sitomaan toteutukseen koulutuksen henkilöt sekä koulutuksen sijainnit (Kuva 6).



Kuva 6. Harjoitusten sitominen aikaan ja paikkaan

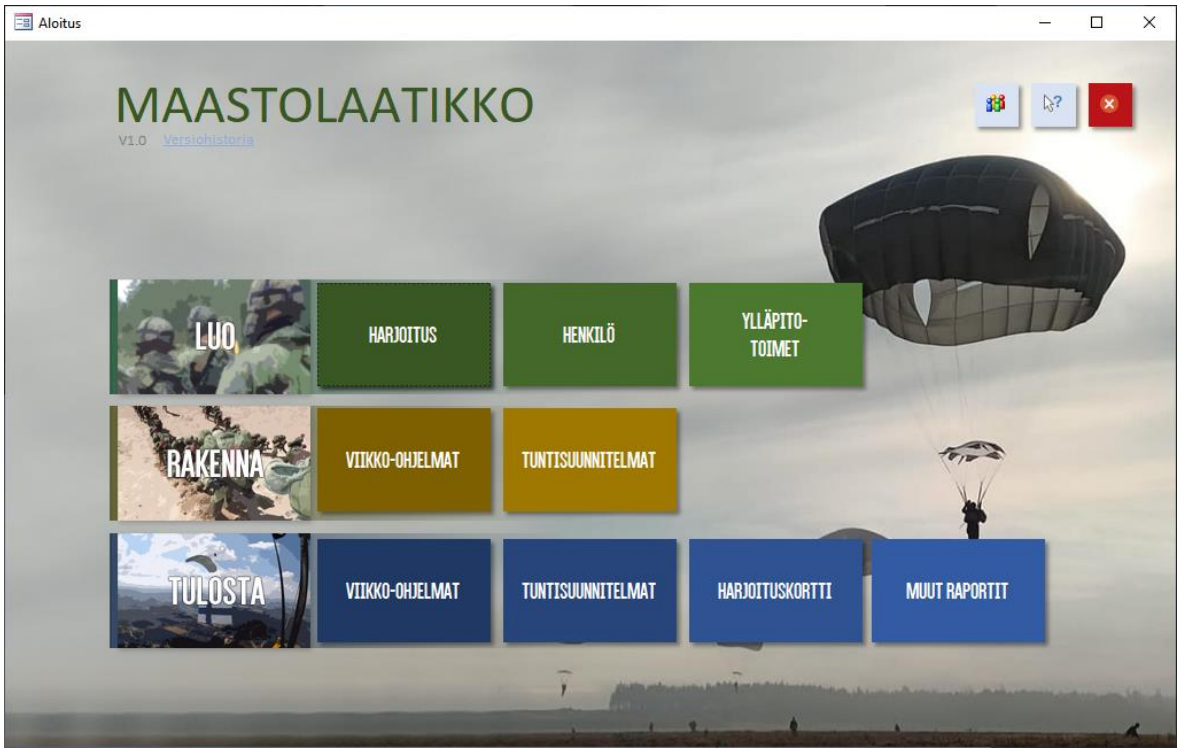
Lomakkeiden toimintoja testatessa havaittiin, että yksittäisten ominaisuuksien lisäämisiin ei tarvitsisi luoda niin useita irrallisia lomakkeita. Tämän pohjalta yhdistettiin sijaintien, oppiaineiden, oikeuksien ja osastojen lisäämiset yhdelle ylläpitolomakkeelle.

4.2.3 Käyttöliittymä

Projektin käyttöliittymä koostuu päävalikosta sekä sovelluksen toimintaan liittyvistä ponnahdusikkunalomakkeista. Sovelluksessa suosittiin paljon ponnahdusikkunoita, sillä varsinkin sovelluksen käyttöönottovaiheessa on käytettävä paljon tukena aikaisemmin suunnitteluun käytettynä taulukkolaskentatiedostoja. Ponnahdusikkunalla saatiin sovelluksen lomakkeista kooltaan pienempiä, jolloin ne veisivät mahdollisimman vähän ruututilaa. Ponnahdusikkunoilla oli myös toinen, jopa tärkeämpi, tarkoitus. Sovelluksen toiminnassa on monta sellaista pistettä, jossa voidaan alkaa rakentamaan uutta osaa ilman, että kaikki vaativat muuttujat on vielä luotu. Esimerkiksi jos viikko-ohjelmaa tehtäessä huomataan, että halutusta harjoituksesta ei ole luotu vielä koulutuskorttia, käyttäjä voi lennosta siirtyä harjoituslomakkeelle luomaan halutun kortin, mikä on sen jälkeen käytettävissä viikko-ohjelman luonnissa. Nämä tilanteet ovat selkeämpiä hallita ponnahdusikkunoilla, sillä alla oleva ikkuna ei mene kiinni ja kun uusi tieto on lisätty, ikkunan voi sulkea lopullisesti eikä se jää tietokantasovellukseen avoimeksi välilehdeksi.

Pelkällä Accessilla on äärimmäisen huonot mahdollisuudet luoda graafisesti ainutlaatuisia käyttöliittymiä. Ohjelmisto keskittyy täysin tietokannan toimintojen toteuttamiseen samalla tavalla kuin Microsoft Word keskittyy tekstinkäsittelyyn. Jotta sovellukseen saataisiin edes vähän mielenkiintoa herättävä alkuvalikko, piti toteuttamisessa turvautua myös Adobe Photoshop -kuvankäsittelytyökaluun. Photoshopilla tehtiin päävalikkoon kaikki muut osat paitsi otsikko ja painikkeet. Taustakuvaan oli luotava lopullinen ryhmittely konseptinmukaisesti, mutta myöhemmin olisi vielä mahdollista päättää painikkeiden määrä. Rajoituksena oli toki, että painikkeen olisi selkeästi sovittava johonkin toimintoryhmään: luo, rakenna tai tulosta.

Projektin tässä vaiheessa oli sovellukselle keksittävä vähintään työnimi. Työnimeksi, joka tälläkin kertaa jäi myös elämään, valittiin ”Maastolaatikko”. Maastolaatikolla tarkoitetaan esimerkiksi ennen hyökkäystä rakennettua pienoismallia, jossa mallinnetaan hyökättävän alueen maasto ja kriittiset kohteet, minkä avulla joukolle käsketään (ohjeistetaan) hyökkäyksen toteutus. Sovellus toimii tällaisen maastolaatikkona, jossa mallinnetaan harjoitus suunnittelemalla sen sisältö ja aikautus, minkä jälkeen siitä tulostetaan koulutuskortti, jolla kouluttajat ohjeistetaan toteuttamaan harjoitus halutulla tavalla. Julkaisuvaiheessa sovelluksen päävalikosta tuli kuvan 7 mukainen.



Kuva 7. Sovelluksen päävalikko

Päävalikkoon luotiin yksikön toimintaan liittyvä taustakuva ja sovelluksessa käytetyt keskeiset värit kuvasivat myös koulutuksessa tapahtuvaa toimintaa. Taustakuva ja kuvassa esiintyvä painikkeiden ryhmittely toteutettiin Adobe Photoshop -kuvankäsittelyohjelmalla. Väreiksi valittiin vihreä ja ruskea kuvaamaan maanläheisyyttä sekä sininen kuvastamaan koulutukseen olennaisesti liittyvää ilmaelementtiä. Valikkorakennetta on tarkoitus lukea niin, että ryhmän otsikko otetaan myös mukaan: painikkeissa esiintyvät otsikot voivat olla samoja, mutta ryhmän otsikko antaa painikkeelle eri merkityksen. Tästä esimerkkinä kaksi ”viikko-ohjelmat” -painiketta, joista toinen on ryhmäotsikko mukaan luettuna ”rakenna viikko-ohjelmat” ja toinen ”tulosta viikko-ohjelmat”.

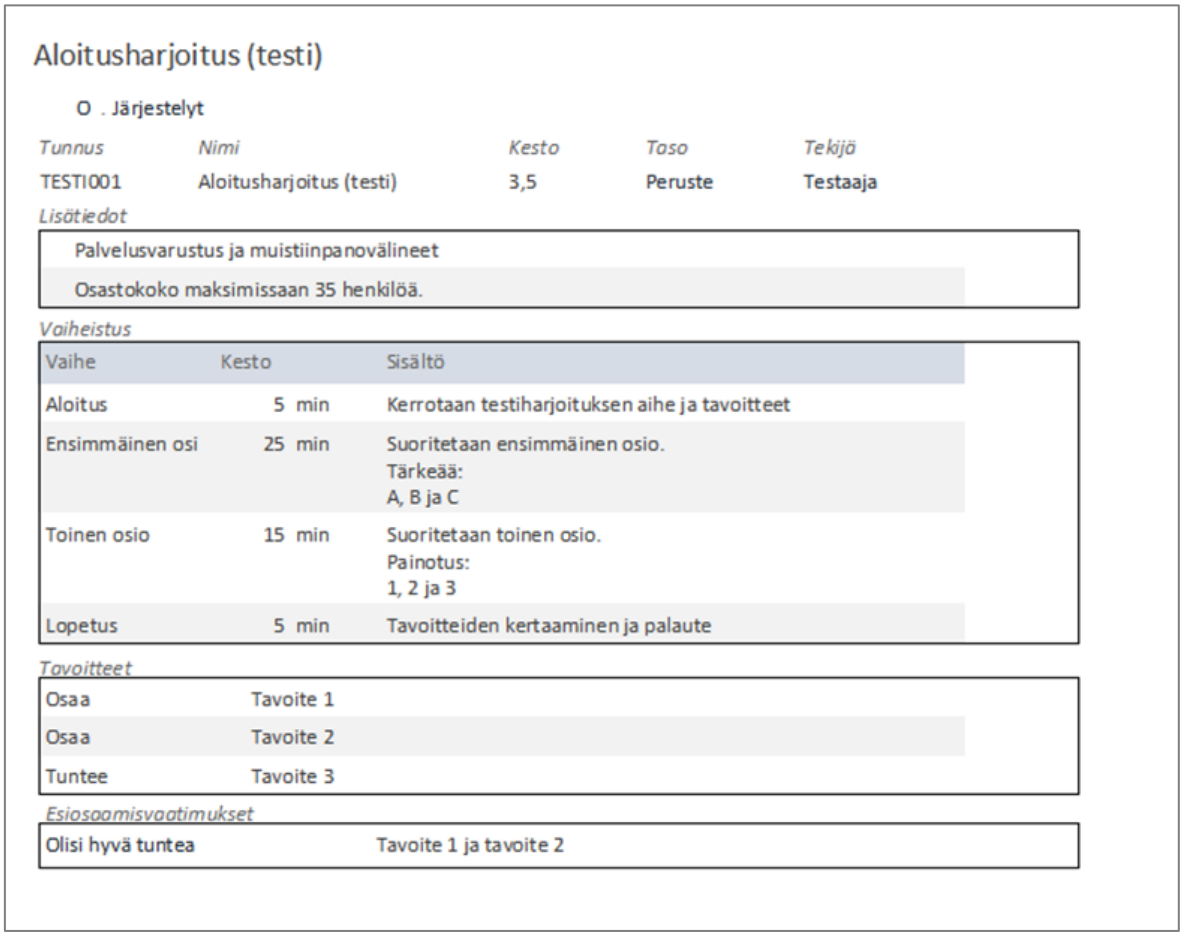
Käyttöliittymän oikeaan yläkulmaan sijoitettiin sovelluksen toimintaan liittyvät painikkeet: kehitysideat, ohjeet sekä sovelluksen sulkeminen. Sommittelu muodostaa verkkosivuille ja sovelluksille tyypillisen ulkoasun, joten käyttäjän on helppo navigoida valikossa.

4.2.4 Raportit

Jos lomakkeet luotiin tiedon syöttämistä varten, niin tietojen tulostamista varten piti luoda määrämuotoisia raportteja. Yleisesti raportit mielletään kaaviokuvia sisältävinä selonteko-objekteina, mutta Access-tietokantatyökalussa niillä tarkoitetaan tietokannasta haluttuun sommitteluun ja ryhmiteltyyn tarkoitettuja tulosteita. Sovelluksen tärkeimmät raportit liittyvät

koulutuskortteihin, viikko-ohjelmiin sekä tuntisuunnitelmiin ja projektisuunnitelman mukainen priorisointi luotiin juuri tässä järjestyksessä.

Koulutuskorttien tulostamisessa tuli ottaa huomioon, että raportti pitää sisällään kaikki siinä haluttavat tiedot, mutta myös, että tiedot mahtuvat ja tulostuvat järkevästi sijoiteltuina. Osa koulutuskorteista mahtui yhdelle sivulle, mutta toiset voivat laajentua jopa 8-sivuisiksi. Tällainen responsiivisuus kyettiin toteuttamaan Access-työkalulla sallimalla kaikkien aliraporttien niin suurentua kuin pienentyäkin. Näin ollen, jos esimerkiksi koulutukseen ei ole esiosaamisvaatimuksia, esiosaamisvaatimus-aliraportti ei tulostu ollenkaan. Esimerkki koulutuskortista on esitelty kuvassa 8.



Aloitusharjoitus (testi)

O . Järjestelyt

Tunnus	*Nimi*	*Kesto*	*Taso*	*Tekijä*
TESTI001	Aloitusharjoitus (testi)	3,5	Peruste	Testaaja

Lisätiedot

Palvelusvarustus ja muistiinpanovälineet
Osastokoko maksimissaan 35 henkilöä.

Vaiheistus

Vaihe	Kesto	Sisältö
Aloitus	5 min	Kerrotaan testiharjoituksen aihe ja tavoitteet
Ensimmäinen osi	25 min	Suoritetaan ensimmäinen osio. Tärkeää: A, B ja C
Toinen osio	15 min	Suoritetaan toinen osio. Painotus: 1, 2 ja 3
Lopetus	5 min	Tavoitteiden kertaaminen ja palaute

Tavoitteet

Osaa	Tavoite 1
Osaa	Tavoite 2
Tuntee	Tavoite 3

Esiosaamisvaatimukset

Olisi hyvä tuntea	Tavoite 1 ja tavoite 2

Kuva 8. Koulutuskortti

Viikko-ohjelmien tulostaminen toteutettiin niin viikon numeron kuin maksujakson mukaan. Maksujaksolla tarkoitetaan työaikakirjanpidollisesti 3 viikon työjaksoa ja sen numerointia. Käyttäjältä kysytään raporttia aukaistaessa esimerkiksi ”Syötä viikon numero” ja syötteen mukaan tietokannasta haetaan kaikki sen kalenteriviikon toteutukset (Kuva 9). Raportti pi-

tää sisällään 3 lajittelutasoa: ensin päivämäärän mukaan nousevasti, sitten kellonajan mukaan nousevasti ja lopuksi henkilöstö niin, että vastuukouluttaja ensimmäisenä. Vastuukouluttaja on korostettu raportissa aktiivisella valintaruudulla.

Viikko-ohjelma (viikko 51) — Fyysinen koulutus 10,5 h

	Harjoitus	Sijainti	Kouluttajat		
ti 20.12.2022			Fyysinen koulutus 2,5 h		
8.00 - 10.00	**Aloitusharjoitus (testi)** *- Aloitusoppitunti, tavoitteet, sisältö, kyselyt, palautteet*	**Kymenlaakso**	Kers Alik	Testaaja Koestaja	☑ ☐
10.00 - 12.30	**Testiharjoitus 2**	**Kymenlaakso**	Alik Kers	Koestaja Testaaja	☑ ☐
14.00 - 16.30	**Testiharjoitus 3**	**Lka Utti** **Lka Vehniäinen**	Kers	Testaaja	☑
ke 21.12.2022			Fyysinen koulutus 4 h		
8.00 - 16.30	**Testiharjoitus 4** *- 1.hyppy: uloshyppy, maahantulo, varjon kokoaminen*	**Takakenttä**	Alik Kers	Koestaja Testaaja	☑ ☐
to 22.12.2022			Fyysinen koulutus 4 h		
8.00 - 12.00	**Laskuvarjohyppy (hyppy 1)**	**Kymenlaakso**	Kers	Testaaja	☑
8.00 - 12.00	**Testiharjoitus 5**	**Kasarmi**	Kers Alik	Testaaja Koestaja	☑ ☐

Kuva 9. Esimerkkiraportti viikko-ohjelmasta

5 Yhteenveto

5.1 Tulokset

Opinnäytetyön tavoitteena oli luoda tietokantapohjainen koulutuksen suunnittelusovellus. Sovellukselle asetettuja tavoitteita olivat yksittäisten koulutusten suunnittelu, valmistelu ja kehittäminen, viikko-ohjelmien tekeminen sekä henkilöstön käytön suunnittelu tuntisuunnitelmien muodossa. Opinnäytetyön toteuttamiseen luotiin projektisuunnitelma ja alustava aikataulu, minkä mukaan sovellus olisi testattu 26.11.2021 mennessä ja pian sen jälkeen otettu alustavasti käyttöön.

Suunnitellun aikataulun mukaisesti saatiin toteutettua suurin osa sovellukselle asetetuista tavoitteista. Sovellus julkaistiin testikäyttöön joulukuussa 2021. Tässä vaiheessa sovelluksessa kyettiin tekemään kaikki tarvittavat ylläpitotoimenpiteet kuten henkilöstön ja sijaintien lisääminen sekä tärkeimpänä koulutustapahtumien suunnittelu, tallentaminen ja tulostaminen. Myös viikko-ohjelmia pystyttiin järjestelmällä luomaan, mutta testikäytön perusteella toiminnossa havaittiin vielä niin paljon kehityskohteita, että järjestelmän jatkokehitykselle ja käyttöönotolle luotiin oma aikataulu (Taulukko 6).

Vaihe	**Ajankohta**	**Teema**
Testikäyttö	12/2021	• Tietoaineiston haaliminen • Puuttuvien tavoitteiden toteuttaminen
Alustava käyttöönotto	6/2022	• Tehostettu käyttö nykyjärjestelmän ohessa • Sovelluksen kehittäminen käyttäjien ehdotusten mukaisesti
Täysi käyttöönotto	6/2023	• Vanha järjestelmä ja sen tuotteet poistuvat käytöstä

Taulukko 6. Käyttöönoton vaiheistus

Vaiheittaisella käyttöönotolla pyrittiin turvaamaan jatkokehitys, lisäämään totuttautumisaikaa järjestelmän käyttöön, mutta kuitenkin tietoaineistoa haluttiin ruveta keräämään mahdollisimman pian. Tämä siksi, että henkilöstön siirtyminen muihin tehtäviin on säännöllistä ja haluttiin taltioida heidän suunnittelutyönsä tulokset, ettei niin kutsuttua hiljaista tietoa siirry pois henkilön mukana. Vuoden 2021 lopussa järjestelmään oli saatu toteutettua jo yli 20 valmista koulutuskorttia, mutta tavoitteena on saada niitä useita satoja ennen lopullista käyttöönottoa.

Jatkokehitysvaiheeseen siirtyi alkuperäisestä suunnitelmasta tuntisuunnitelmien ja vuosisuunnitelmien tekeminen sekä osa raportointimahdollisuuksista. Näiden lisäksi järjestelmää kehitetään jatkuvasti esiin nousseiden kehitysehdotusten mukaisesti. Kehitysehdotukset käsitellään työyksikön normaalin palaverikäytäntöjen mukaisesti.

5.2 Pohdinta

Ohjelmistoprojektin onnistumiseen vaikuttaa erittäin paljon se, miten hyvin sen suunnittelu onnistuu. Projektin suunnittelu oli onnistunut riittävän hyvin, sillä vaikka projektin aikana ilmeni toistuvasti muutostarpeita suunnitelmaan, yksikään muutos ei kuitenkaan paljastunut kohtalokkaaksi. Selkeimmin tämä oli todettavissa tietokannan osalta: attribuutteja jouduttiin lisäilemään, muuttelemaan ja poistelemaan jälkikäteen, mutta muutokset eivät aiheuttaneet ketjureaktioita koko tietokantaan ja siinä esiintyviin yhteyksiin.

Projektin aikataulun suunnittelu olisi voinut olla yksityiskohtaisempaa. Vaikka se oli nyt luotu alustavaksi ja suuntaa antavaksi, se helpotti projektin toteuttamista, kun oli jokin runko toteutukselle. Tarkemmalla suunnittelulla olisi voitu aikatauluttaa tehtäviä paremmin ja vähentää poukkoilua vaiheesta toiseen. Toisaalta suunnittelu koettiin vähäisen kokemuksen takia haastavaksi. Selkeäksi puutteeksi suunnittelussa ilmeni se, ettei opinnäytetyöraporttia otettu täydennettäväksi sovelluskehityksen ohessa. Kun raportointi oli ajoitettu projektin päätteeksi, oli lähes mahdotonta muistaa havaittuja ongelmia ja niihin keksittyjä ratkaisuja.

Access -tietokantatyökalun käyttö lopputyön toteuttamisessa aiheutti alkuun paljon haasteita. Tutkinnon aikana ehdittiin hyvin perehtyä selainpohjaisiin toteutuksiin useilla eri ohjelmointikielillä, mutta tätä varten jouduttiin opiskelemaan jälleen uusi tapa toteuttaa tietokantasovellus. Onneksi tietokantojen perusteet olivat hyvin hallussa, joten harjoiteltavaksi jäi vain se, miten kaikki toiminnot toteutetaan tällä kyseisellä työkalulla. Useisiin eri ohjelmointikieliin ja toteutustapoihin tutustuminen tutkinnon aikana nopeuttivat tämän työkalun opettelemista. Insinöörille keskeinen taito on löytää ratkaisu ongelmaan mitä tahansa tapaa käyttäen, joten työkalupakissa on hyvä olla monta eri työkalua.

Työssä pyrittiin korvaamaan kokonainen toimintatapa eli järjestelmä. Käytössä olevan järjestelmän tai menetelmän korvaaminen uudella ei koskaan ole tunteeton tapahtuma: muutokseen reagoidaan aina ja huolellinen suunnitelma muutoksen toteuttamisesta luo edellytyksen siihen, että reaktio on positiivinen. On sanomattakin selvää, että pitkään käytössä ollut tapa tehdä suunnitelmia on vaatinut nykyisiltä kouluttajilta aikaa ja panostusta sen käyttämiseen. Muutoksen markkinoinnissa tulisikin varautua pahimpaan vaihtoehtoon eli heihin, jotka ovat juuri saaneet opeteltua nykyisen järjestelmän tehokkaan käytön eikä näin ollen ole varmasti riemuissaan järjestelmän korvaamisesta. Muutosvastarintaa voi lievittää

painottamalla uuden järjestelmän käyttöönotosta aiheutuvia hyötyjä sekä pyrkiä rakentamaan sovelluksesta mahdollisimman helppokäyttöinen; mahdollisuuksien mukaan jopa helpompikäyttöinen kuin jo opittu, käytössä oleva järjestelmä. Sovelluksen käyttöönoton suunnitelma jäi tämän työn osalta puutteelliseksi, mutta se lisättiin jatkokehitykseen yhdeksi selkeäksi painopisteeksi. Muutosjohtamisessa on otettava huomioon se, että käyttöönoton alkuvaiheessa työmäärä hetkellisesti jopa kasvaa, kun sovellusta käytetään nykyjärjestelmän ohella. Työyksikön toimintaa voitaisiin luonnehtia yksinkertaistettuna niin, että saapumiserän koulutusta on toteutettu osin projektinomaisesti, vaikka sitä voitaisiin pääosin käsitellä prosessinomaisesti.

Opinnäytetyön aiheen valinta ja työn rajaus onnistuivat hyvin. Lopputuloksena syntynyt sovellus ehdittiin ottaa riittäviltä osin käyttöön ja se edistää kohdeorganisaation toimintaa hyvin. Sovellus antaa selkeän mahdollisuuden yhtenäistää kunkin vuosikurssin koulutusta, jolloin vuosittaista suunnittelutyötä voidaan systemaattisesti vähentää.

Lähteet

Ammouri, YM. 2015. Agile Software Development Basics and fundamentals. Code Project. Viitattu 31.10.2021. Saatavissa https://www.codeproject.com/Articles/1064114/Agile-Software-Development-Basics

Babish, N. 2019. Using Red and Green in UI Design. Viitattu 26.11.2021. Saatavissa https://uxplanet.org/using-red-and-green-in-ui-design-66b39e13de91

Chand, M. 2021. What is a Relational Database. C# Corner. Viitattu 17.11.2021. Saatavissa https://www.c-sharpcorner.com/article/what-is-a-relational-database/

Date, C. 2013. Relational Theory for Computer Professionals. Chapter 1. Basic Database Concepts. O'Reilly Media Inc. ISBN: 9781449369439. Viitattu 10.11.2021. Saatavissa https://www.oreilly.com/library/view/relational-theory-for/9781449365431/ch01.html

Demchenko, M. 2021. Software Development Life Cycle: A Guide to Phases and Models. Ncube. Viitattu 28.10.2021. Saatavissa https://ncube.com/blog/software-development-life-cycle-guide.

Dunleavy, J., Gallagher A. & Reeves P. 2019. The Waterfall Model: Advantages, disadvantages, and when you should use it. IBM Developer. Viitattu 28.10.2021. Saatavissa https://developer.ibm.com/articles/waterfall-model-advantages-disadvantages/

Enge, E. 2021. Mobile vs. Desktop Usage in 2020. Perficient. Viitattu 22.11.2021. Saatavissa https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage

Erikoisjääkäripataljoona 2016. Erikoisjääkäripataljoonan oma esittelymateriaali. Viitattu 15.11.2021.

Eriksen 2020. How Translating Colors Across Cultures Can Help You Make a Positive Impact. Viitattu 26.11.2021. Saatavissa https://eriksen.com/marketing/color_culture/

Friedman, V. 2018. Responsive Web Design - What It Is And How To Use It. Smashing Magazine. Viitattu 16.11.2021. Saatavissa https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/

Hannah, J. 2021. An Introduction to Color Theory and Color Palettes. CareerFoundry. Viitattu 26.11.2021. Saatavissa https://careerfoundry.com/en/blog/ui-design/introduction-to-color-theory-and-color-palettes/

Harness 2021. Understanding the Phases of the Software Development Life Cycle. Harness Author. Viitattu 24.10.2021. Saatavissa https://harness.io/blog/software-development-life-cycle/

Helsingin kaupunki 2021. Vaatimukset (perinteinen hanke). Viitattu 09.11.2021. Saatavissa https://kehmet.hel.fi/menetelmalaari/vaatimukset/

Indeed 2021. What Is a User Interface? (Definition, Types and Examples). Indeed Editorial Team. Viitattu 14.11.2021. Saatavissa https://www.indeed.com/career-advice/career-development/user-interface

Intti.fi 2022. Ennen palvelusta haettavat palvelustehtävät – Erikoishaku. Viitattu 11.11.2021. Saatavissa https://www.intti.fi/erikoisjoukot

JHS 173 2018. JHS 173 ICT-palvelujen kehittäminen: Vaatimusmäärittely. JUHTA – Julkisen hallinnon tietohallinnon neuvottelukunta. Viitattu 09.11.2021. Saatavissa https://www.suomidigi.fi/ohjeet-ja-tuki/jhs-suositukset/jhs-173-ict-palvelujen-kehittaminen-vaatimusmaarittely-vanhentunut

Keerti. 2020. A Complete Guide for UI Design Process (User Interface). CronJ. Viitattu 17.11.2021. Saatavissa https://www.cronj.com/blog/user-interface-ui-design-process-in-graphic-design/

Laskuvarjojääkärikomppania 2022. Laskuvarjojääkärikomppanian oma esittelymateriaali. Viitattu 23.11.2021.

Majewski, M. 2019. Top 6 Software Development Methodologies. Planview Blog. Viitattu 28.10.2021. Saatavissa https://blog.planview.com/top-6-software-development-methodologies/

Masters In Data Science 2020. What is the difference between UX and UI?. 2U Inc. Viitattu 19.11.2021. Saatavissa https://www.mastersindatascience.org/learning/difference-between-ui-and-ux/

Murphy, C. 2019. Fundamentals of UI Design – Part 1. Adobe. Viitattu 13.11.2021. Saatavissa https://xd.adobe.com/ideas/guides/fundamentals-ui-design-part-1/

Peterson, R. 2021. What is Normalization in DBMS (SQL)? 1NF, 2NF, 3NF, BCNF Database with Example. Viitattu 12.11.2021. Saavilla https://www.guru99.com/database-normalization.html

Preston, M. 2021. System Development Life Cycle Guide. CloudDefense. Viitattu 24.10.2021. Saatavissa https://www.clouddefense.ai/blog/system-development-life-cycle

Puolustusvoimat 2021. Tietoa meistä. Puolustusvoimat. Viitattu 7.10.2021. Saatavissa https://puolustusvoimat.fi/tietoa-meista

Ribgy, D., Sutherland, J. & Takeuchi, H. 2021. Embracing Agile: How to master the process that's transforming management. Harvard Business Review. Viitattu 31.10.2021. Saatavissa https://hbr.org/2016/05/embracing-agile

Sacolick, I. 2020. What is agile methodology? Modern software development explained. InfoWorld. Viitattu 02.11.2021. Saatavissa https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html

Sharma, O. 2019. Introduction To Databases. C# Corner. Viitattu 10.11.2021. Saatavissa https://www.c-sharpcorner.com/article/introduction-to-databases/

Slawek-Polczynska, A. 2020. Is Agile always the best solution for software development projects? SolDevelo. Viitattu 14.11.2021. Saatavissa https://www.soldevelo.com/blog/is-agile-always-the-best-solution-for-software-development-projects/

SPD Load 2019. UI vs UX: What is UI? What is UX? And What's the Difference? Viitattu 19.11.2021. Saatavissa https://spdload.com/blog/ux-vs-ui-design/

Stiner, S. 2016. Using The Five Stages Of The Software Development Process To Spur Innovation. Forbes Technology Council. Viitattu 25.10.2021. Saatavissa https://www.forbes.com/sites/forbestechcouncil/2016/09/14/using-the-five-stages-of-the-software-development-process-to-spur-innovation/?sh=d82042c5ba03

Suomidigi 2021. JHS-suositukset ja VAHTI-ohjeet. Viitattu 10.11.2021. Saatavissa https://www.suomidigi.fi/ohjeet-ja-tuki

US Department of Justice. 2003. The Department of Justice Systems Development Life Cycle Guidance Document. Viitattu 25.10.2021. Saatavissa https://www.justice.gov/archive/jmd/irm/lifecycle/table.htm

Vaniukov, S. 2020. Colors in UI Design: A Guide for Creating the Perfect UI. Usabilitygeek. Viitattu 18.11.2021. Saatavissa https://usabilitygeek.com/colors-in-ui-design-a-guide-for-creating-the-perfect-ui/

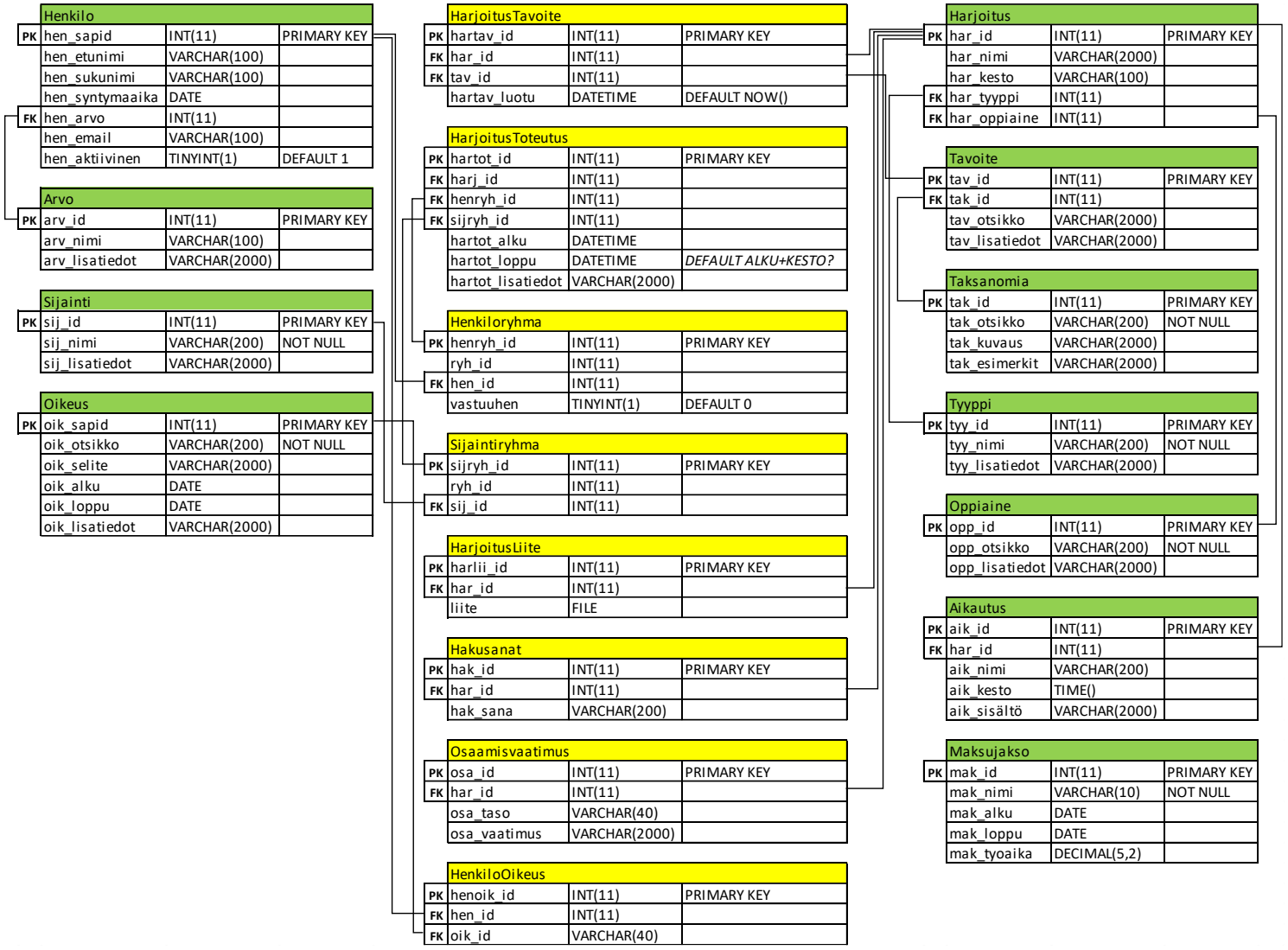
Liitteet

Jakson tuntimäärä	114,45
Jakson alku pvm	4.10.2021
Maksujakson numero	15

Arvo ja nimi	Viikko 40							Viikko 41				
	Ma	Ti	Ke	To	Pe	La	Su	Ma	Ti	Ke	To	Pe
	4.10.	5.10.	6.10.	7.10.	8.10.	9.10.	10.10.	11.10.	12.10.	13.10.	14.10.	15.1
	TEEMA 1			TEEMA 2				TEEMA 3				
HENKILÖ 1	7:39	7:39	7:39	7:39	7:39	VL	VL	7:39	7:39	7:39	7:39	7:3
HENKILÖ 2	7:39	7:39	7:39	7:39	7:39	VL	VL	7:39	7:39	7:39	7:39	7:3
HENKILÖ 3	7:39	7:39	7:39	7:39	7:39	VL	VL	7:39	7:39	7:39	7:39	7:3
HENKILÖ 4	7:39	7:39	7:39	7:39	7:39	VL	VL	7:39	7:39	7:39	7:39	7:3

PUOLUSTUSVOIMAT	**Viikko-ohjelma**		
UTIN JÄÄKÄRIRYKMENTTI	**Sunnuntai 10.10.**		
Laskuvarjojääkärikomppania **Viikko 40**	**VLV**		
Maanantai 04.10.	**Tiistai 05.10.**	**Keskiviikko 06.10.**	**Tors**
08.00 - 11.30 **TYYPPI+OPPIAINE** Otsikko - sisältö **Paikka** **Kouluttajat** **13.00 - 16.30** **TYYPPI+OPPIAINE** Otsikko - sisältö **Paikka** **Kouluttajat**	**08.00 - 11.30** **TYYPPI+OPPIAINE** Otsikko - sisältö **Paikka** **Kouluttajat** **13.00 - 16.30** **TYYPPI+OPPIAINE** Otsikko - sisältö **Paikka** **Kouluttajat**	**08.00 - 11.30** **TYYPPI+OPPIAINE** Otsikko - sisältö **Paikka** **Kouluttajat** **13.00 - 16.30** **TYYPPI+OPPIAINE** Otsikko - sisältö **Paikka** **Kouluttajat**	**08.00 - 11.30** **TYYPPI+OPPIA** Otsikko - sisältö **Paikka** **Kouluttajat** **13.00 - 16.30** **TYYPPI+OPPIA** Otsikko - sisältö **Paikka** **Kouluttajat**
Fyysinen koulutus: 8	*Fyysinen koulutus:* 4	*Fyysinen koulutus:* 4	*Fyysinen koulu* 1
Viikon tavoitteet:		**Vastuukouluttajat:**	**Paikka ja aika:** **UTTI**

Liite 1. Esimerkki tuntisuunnitelmasta ja viikko-ohjelmasta.



Liite 2. Sovelluksen tietokantamalli