Ohjelmistoratkaisu kenttäoperaattoreille toistuvien tehtävien tueksi

Datankeräyksen digitalisointi ja sen mahdolliset hyödyt

Lappeenrannan-Lahden teknillinen yliopisto LUT

Tietotekniikan kandidaatintyö

2025

Jarno Forsblom

Tarkastaja: Associate Professor Ari Happonen

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Sciences

Software Engineering

Jarno Forsblom

Software solution for repetitive field operations

Digitalization of data collection and its potential benefits

Bachelor's thesis

28 pages, 2 figures and 2 tables

Examiner: Associate Professor Ari Happonen

Keywords: industrial revolution 4.0, field operation, digitization, data collection, software production, case study

The Industrial Revolution 4.0 emphasizes the importance of data in process control. Paper-based data collection can be inefficient and error-prone, whereas digital solutions can improve accuracy but require careful consideration of usability.

The purpose of this bachelor's thesis was to explore alternatives to a software solution for Hiava Oy to digitize the recurring collection of customer data in field operations. A case study and literature review were employed as research methods to examine key software development challenges and existing systems

A requirement specification was created for Hiava Oy , and a prototype-based UX testing approach was applied in the implementation to ensure usability was considered throughout the development process. The software solution was built using React and Node.js technologies, enabling the same programming language to be applied across both the frontend and backend.

The resulting solution provides a model for digitizing customer data within the company's operating environment, though further research is required in areas such as information security and system integration.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT Teknis-luonnontieteellinen

Tietotekniikan koulutusohjelma

Jarno Forsblom

Ohjelmistoratkaisu kenttäoperaattoreille toistuvien tehtävien tueksi

Datankeräyksen digitalisointi ja sen mahdolliset hyödyt

Tietotekniikan kandidaatintyö

28 sivua, 2 kuvaa ja 2 taulukkoa

Tarkastaja: Associate Professor Ari Happonen

Avainsanat: teollinen vallankumous 4.0, kenttätyö, digitalisointi, datakeräys, ohjelmistotuotanto, case-tutkimus, tapaustutkimus

Teollinen vallankumous 4.0 korostaa datan merkitystä prosessien ohjauksessa. Paperimuotoinen tiedonkeruu voi olla tehotonta ja virhealtista, kun taas digitaalinen ratkaisu voi olla tarkempi, mutta edellyttää käytettävyyden huomioimista.

Tämän kandityön tarkoituksena oli tutkia vaihtoehtoja ohjelmistoratkaisulle, joka digitalisoi Hiava Oy:n kenttäoperaatioissa toistuvan asiakasdatan keräyksen. Tutkimusmenetelminä käytettiin tapaustutkimusta ja kirjallisuuskatsausta, joiden avulla tarkasteltiin ohjelmistokehityksen haasteita ja olemassa olevia järjestelmiä.

Hiava Oy:lle luotiin vaatimusmäärittely ja toteutuksessa sovellettiin prototyyppipohjaisen UX-testauksen lähestymistapaa, jota hyödyntämällä voitiin huomioida käytettävyys osana kehitystä. Ohjelmistoratkaisu rakennettiin React- ja Node.js-teknologioilla, joka mahdollistaa saman ohjelmointikielen hyödyntämisen front- ja backend ratkaisuissa.

Tutkimuksen pohjautuva ohjelmistoratkaisu tarjoaa mallin asiakasdatan digitalisointiin yrityksen toimintaympäristössä, mutta lisätutkimusta tarvitaan esimerkiksi tietoturvan ja järjestelmäintegraation osalta.

KIITOKSET

Haluan kiittää Hiava Oy:tä mielenkiintoisen aiheen tarjoamisesta. Kiitos työn ohjaajana toimineelle Ari Happoselle avusta ja ohjauksesta.

Lyhenneluettelo

API – Application Programming Interface (Ohjelmointirajapinta)

Backend – palvelinosa, joka käsittelee pyynnöt, logiikan ja tiedonsiirron

CORS – Cross-Origin Resource Sharing

CSS – Cascading Style Sheets

Frontend – käyttäjän kanssa vuorovaikutuksessa oleva osa ohjelmasta

JSON – JavaScript Object Notation

JWT – JSON Web Token

Mock/mockup – malli tai jäljitelmä, jolla voidaan testata toiminnallisuutta

REST – Representational State Transfer (Tilaohjattu arkkitehtuuri)

UI – käyttöliittymä

UX – käyttäjäkokemus

Sisällysluettelo

Abstract

Tiivistelmä

Kiitokset

Lyhenneluettelo

1 Johdanto 7
1.1 Paperilta digitaaliseen datankeräykseen siirtyminen 7
1.2 Käytettävyyden huomiointi osana kehitystä 8
1.3 Tavoitteet ja rajaukset 10
2 Tutkimusmenetelmät 11
2.1 Tutkimuksen tyyppi 11
3 Ohjelmistoratkaisun toteutuksesta 13
3.1 UI/UX suunnittelu 13
3.2 Datan käsittely ja siirtäminen 14
3.3 Digitaalinen datankeräys kenttätyössä 15
3.3.1 Valmiit kaupalliset ratkaisut 15
3.3.2 Avoimen lähdekoodin esimerkki 17
4 Tapaus: Hiava Oy:n kenttädatankeräys 18
4.1 Vaatimusmäärittely 18
4.2 Teknisten ratkaisujen valinta 20
4.2.1 Frontend-arkkitehtuurin suunnittelu 20
4.2.2 Backend testiarkkitehtuurin suunnittelu 21
4.2.3 Roolipohjainen käyttäjäoikeus 22
4.3 Käyttöliittymän toteutus 22
4.3.1 Dynaaminen lomakkeiden luonti 23
4.3.2 Lomakkeiden täyttäminen 25
5 Keskustelu 26
6 Yhteenveto 28
Lähteet 29

1 Johdanto

Teollinen vallankumous 4.0 tarkoittaa digitaalisten teknologioiden laaja-alaista hyödyntämistä yritysten tuotannossa ja arvonluontiketjussa. Sen perustana toimii tehokas datan kerääminen, analysointi ja hyödyntäminen tuotantoprosessien ohjaamiseen. (Rymarczyk 2020) Teknologisten ratkaisuiden hyödyntäminen voi tarjota yritykselle useita keinoja parantaa tehokkuutta. Näitä voiva olla muun muassa tuotesuunnittelun tehostaminen tai kierrätysmateriaalien ja yrityksen resurssien tehokkaampi hyödyntäminen tuotannossa. Tuotannon optimointi ja tarkempi resurssien hyödyntäminen voivat osaltaan myös vähentää yrityksen päästöjä (Happonen ja Ghoreishi 2022).

Tässä työssä selvitettiin, kuinka Hiava OY:n suorittamassa kenttätyössä tapahtuvaa ja prosessien ohjauksessa hyödynnettävää datankeräystä voidaan tehostaa ja parantaa digitalisoimalla keräysprosessia. Kerättävän datan tarkoituksena on suorittaa prosessiin ja toiminnalliseen työmäärään pohjautuva työnkulun johtaminen ja työntekijöiden allokointi. Tavoitteena oli luoda korkean käytettävyyden ohjelmistoratkaisu kenttäoperaattoreiden suorittamien tehtävien tueksi.

1.1 Paperilta digitaaliseen datankeräykseen siirtyminen

Paperilla kerättävällä datalla on luonnollisesti omat haasteensa, kuten kerätyn datan siirtäminen ja käyttäminen prosessien ohjauksessa, joka vaatii datan kaksinkertaista kirjoittamista, ensin paperille ja myöhemmin prosessinohjausohjelmistoon. Tämä luo prosessiin tehottomuutta ja altistaa prosessin kahdesti saman tyyppisille ihmisen tekemille virheille. Toisaalta digitaalisessa datankeräyksessä on omat haasteensa, kuten laitteiden hukkuminen tai hajoaminen, tiedonsiirrossa tapahtuvat virheet, datamenetys ja mahdollinen tietoliikenteen turvaaminen. Myös vapaan tekstin kirjoittaminen tekstilaatikkoon on käyttäjien mukaan helpompaa paperimuodossa kuin pienen digitaalisen laitteen, kuten älypuhelimen, näppäimistöllä (Fletcher et al. 2003; Burgdorf 2016). Kortelaisen, Happosen ja Kinnusen (2016) mukaan nämä käytännön haasteet voivat tehdä perinteisten fyysisten tuotteiden valmistajien siirtymisen kohti digitaalisesti tehostettuja palvelutuotteita lähes mahdottomaksi, mikä hidastaa toimialojen digitalisoitumista.

Paperilla suoritettava datankeräys voi olla myös ketterämpää ja helpommin mukautuvaa, jos ohjelmistoa ei ole rakennettu mahdollisia datankeräyksessä tapahtuvia ja lomakkeen vaatimia muutoksia silmällä pitäen. Toisaalta digitaalinen datankeräys mahdollistaa nopeamman datan käsittelyn ja tulosten saamisen paperiseen prosessointiin liittyvän datan kaksinkertaistamisen vuoksi. Digitaalinen data voidaan halutessa lähettää suoraan keräyksen päätteeksi tai jo sen aikana automaattisesti käsittelyyn ja analysoitavaksi. Ohjelmiston suunnittelun lisäksi onkin oleellista tiedostaa datan keräämiseen liittyvät vaatimukset, ja minkälaiselle työkalulle ohjelmistoa halutaan pääasiallisesti suunnitella.

1.2 Käytettävyyden huomiointi osana kehitystä

Pelkkä lomakkeiden digitalisointi ei riitä, vaan käytettävien työkalujen tulee olla myös käyttäjäystävällisiä ja tukea työprosessia. Tätä tavoitetta voidaan määrittää käytettävyydellä, jonka tarkoitus on kuvata ohjelmiston helppokäyttöisyyttä suhteessa tavoitteen saavuttamiseen. Weichbroth (2024) määrittelee käytettävyyden neljä tärkeintä pääattribuuttia: tarkkuus, suorituskyky, tyytyväisyys ja oppimiskyky. Arditon et al. (2014) mukaan yritysten antamissa ohjelmistomäärityksissä keskitytään laajalti ohjelmistojen toiminnallisuuksiin, mutta jopa neljännes vastaajista jättää määrittelyissään käytettävyyden kokonaan huomiotta. Syitä tälle ovat muun muassa resurssien säästäminen, se, ettei käyttäjäkokemukseen investoinnista nähdä saatavan selkeitä etuja, tai se, ettei aiheesta olla tietoisia. Tämän lopputuloksena voi olla vaikeaselkoinen käyttöliittymä, joka hidastaa työntekoa ja aiheuttaa virheitä, koska ohjelmistojen tuottajat eivät käytä resursseja määrittelyn ulkopuolisiin ominaisuuksiin. Kyseisten ongelmien välttämiseksi onkin oleellista, että ohjelmiston tilaaja kykenee esittämään selkeät ja johdonmukaiset vaatimukset toiminnallisuuksien lisäksi myös käytettävyydelle, jota voidaan testata kehittämisen aikana.

Happosen ja Siljanderin (2020) mukaan käytettävyyteen voidaan panostaa myös gainsharing-mallin kautta, jolloin käytettävyys voidaan nähdä lisäarvoa tuottavana investointina. Tämä perustuu tilaajan ja toimittajan yhteistyöhön, jonka tavoitteena on tuottaa hyötyä molemmille osapuolille. Oleellinen osa tällaisen mallin toimivuutta on luottamus, joka puolesta asiakkaat arvostavat etenkin kehitysprosessin läpinäkyvyyttä. Lisäksi toimittajan on tärkeää pystyä ilmaisemaan selkeästi yhteistyön kautta saatavat taloudelliset säästöt ja hyödyt. Tällaisessa gainsharing-mallissa on tärkeää, että yhteistyön

tulokset jaetaan reilusti molempien osapuolten kesken. Tämä korostuu erityisesti digitalisaatioprosesseissa, joissa jokaisen osapuolen on tehtävä muutoksia omiin toimintatapoihinsa tai järjestelmiinsä. Reilu yhteistyö tukee sitoutumista ja edistää muutoksen onnistumista.

Tällaisessa gainsharing-mallissa on myös tärkeää, että yhteisen panostamisen hedelmät jaetaan reilusti molempien osapuolten kesken. Tämä korostuu erityisesti tilanteissa, joissa siirrytään paperimuotoisesta datankeräyksestä digitaaliseen muotoon, sillä tällaiset muutokset edellyttävät usein jokaiselta prosessiin osallistujalta mukautumista – joko omiin toimintatapoihin tai järjestelmiin. Reilu ja avoin yhteistyö tukee digitalisaation onnistumista ja vahvistaa osapuolten sitoutumista yhteisiin tavoitteisiin.

Myös kehityskustannusten kannalta on tärkeää, että käytettävyyttä ajatellaan jo suunnitteluvaiheessa. Boehmin ja Papaccionin (1988) mukaan mitä aikaisemmassa vaiheessa ongelmat todetaan ja mitä nopeammin niihin reagoidaan, sitä halvemmaksi ongelmien korjaaminen tulee. Pahimmillaan ohjelmiston korjauksesta aiheutuva lasku voi olla kymmeniä tai satoja kertoja suurempi kuin aikaisessa vaiheessa testauksen kautta todetun ongelmakohdan muuttaminen. Näitä kustannuksia voivat paisuttaa muun muassa ohjelmiston uudelleen kirjoittamisesta johtuva tarve lisätestaukselle tai korjatuista osista riippuvien muiden osien toiminnallisuuksien rikkoutuminen.

Kortumin ja Sorberin (2015) mukaan pienemmille älypuhelimen näytöille luoduissa käyttöliittymissä on käyttäjien mukaan parempi käytettävyys kuin tablettitietokoneille luoduissa käyttöliittymissä. Syitä tähän voivat olla pienempikokoisten älypuhelimien rajallinen ruututila, joka pakottaa tekijän rajoittamaan toiminnallisuuden ja sisällön olennaisimpiin ominaisuuksiin. Lisäksi Kortum ja Sorber huomioivat, että erityisesti tabletille luodut käyttöliittymät ovat ulkoasultaan lähempänä web-sivuja, kuin älypuhelimelle luotuja käyttöliittymiä. Tabletille luotu käyttöliittymä ja suurempi näyttö saa kehittäjän usein esittämään enemmän sisältöä ja toiminnallisuuksia kuten itsevieriviä ikkunoita, videoita ja monimutkaisia valikoita. Tämä suurempi elementtien määrä lisää käyttäjän kognitiivista kuormaa, koska tarkkailtavaa tietoa sekä päätöksentekoa on enemmän, mikä voi heikentää koettua käytettävyyttä. Tämän tyyppinen käyttäjän kuormittaminen voisi olla mahdollisesti vältettävissä, jos älypuhelimen käyttöliittymä siirretään sellaisenaan tablettitietkoneelle. Käyttäjät voivat olla myös tottuneempia käyttämään puhelimen ruutua, joka voi osaltaan parantaa niiden koettua käytettävyyttä.

Ozok et al (2008) puolestaan vertailivat käyttäjäkokemuksia tablettitietokoneiden ja kannettavien tietokoneiden välillä eri tehtävissä ja löysivät käyttäjien tekevän suuremman määrän kirjoitusvirheitä tableteilla kirjoittaessa. Tämä oli erityisen huomattavaa vapaata tekstiä tuotettaessa ja tablettien ongelmissa tunnistaa magneettisella kynällä tabletin pintaan käsin kirjoitettua tekstiä. Myös valmiiden lomakkeiden täyttämisessä ilmeni tableteilla suurempi määrä virheitä kuin verrannollisesti kannettavalla tietokoneella täytettävissä lomakkeissa.

1.3 Tavoitteet ja rajaukset

Tässä työssä oli tavoitteena luoda Hiava OY:lle kenttätyöhön käytettävä ohjelmisto, jonka avulla siirrytään paperisesta datankeräyksestä digitaaliseen muotoon. Tavoitteeseen pyrittiin suunnittelemalla ohjelmisto vastaamaan käyttäjien vaatimuksia, tunnistamalla digitaaliseen datankeräykseen liittyviä tunnettuja ongelmia sekä etsimällä niihin tarkoituksenmukaisia ratkaisuja. Ohjelmiston suunnittelussa käyttäjäryhminä olivat vaihtelevalla teknisellä tasolla olevat yrityksen asiakkaat, sekä ohjelmiston toiminnasta ja lomakkeiden ylläpitämisestä vastaavat asiantuntijat.

Datankeräyksen digitalisoinnin lisäksi olennainen osa työtä olivat UX- ja UI-suunnittelu, joiden kautta voitiin mahdollistaa sujuvan ja käyttäjäystävällisen työskentelyn. Vaatimusten ja kenttätyöhön liittyvien tarpeiden sekä rajoitusten pohjalta pyrittiin löytämään parhaaksi katsotut työkalut ja luomaan niitä hyödyntäen helppokäyttöinen ohjelmisto. Käytettävyyden arvioinnissa painotettiin erityisesti hyödyllisyyttä, suorituskykyä, käyttäjätyytyväisyyttä ja oppimiskykyä. Ohjelmistoratkaisun suunnittelussa huomioitiin sekä työprosessin sujuvuus että datankeräykseen liittyvät haasteet. Tarkka vaatimusten määrittely oli tärkeää, jotta voitiin esittää selkeitä ratkaisuvaihtoehtoja ongelmien minimoimiseksi tai välttämiseksi. Määrittelyn avulla pystyttiin myös ennakoimaan mahdollisia ongelmakohtia ja löytämään ratkaisuja suunnittelun aikana esiin nousseisiin haasteisiin. Työ rajattiin koskemaan Hiava OY:n tarpeita ja keskittyi yhden, määrittelyjen perusteella parhaaksi todetun ja toteuttamiskelpoisen alustan kehittämiseen. Työn tutkimuskysymyksiä olivat:

1. Minkälaisia tietoja Hiava OY asiakasdataan kerätään?
2. Millaisia resurssirajoitteita eri ratkaisut asettavat?
3. Mikä on paras ohjelmistoratkaisu Hiava OY:n tarpeisiin?

2 Tutkimusmenetelmät

Tässä osiossa käydään tarkemmin läpi kenttäoperaattoreiden tarpeisiin luotavan ohjelmistoratkaisun tutkimusmenetelmiä. Pääasiallisena työssä käytettävänä tutkimusmenetelmänä toimii tapaustutkimus, jonka avulla tarkastellaan ja kehitetään Hiava Oy:n toimintamalliin sopiva ohjelmistoratkaisu. Tapaustutkimuksen tukena hyödynnetään narratiivista kirjallisuuskatsausta, jonka tarkoitus on koota aiempaa tutkimustietoa ohjelmistokehityksestä. Kenttätyöhön soveltuvan ohjelmiston suunnittelu vaatii kontekstista riippuvaa tarpeiden huomioimista. Tämän vuoksi tapaustutkimus ja sitä tukeva kirjallisuuskatsaus muodostavat yhdessä perustan, jonka varaan ohjelmistoratkaisun kehitys rakennettiin.

2.1 Tutkimuksen tyyppi

Tämän työn tutkimusmenetelmänä hyödynnetään tapaustutkimusta. Tapaustutkimus on empiirinen menetelmä, jonka avulla tarkastellaan ajankohtaista ilmiötä sen todellisessa kontekstissa. Tapaustutkimukset voidaan tarkemmin lajitella vielä kolmeen eri alaryhmään; kysely, koe ja toiminnallinen tutkimus, joista toiminnallisen tutkimuksen tarkoitus on muokata jotain tutkittavan kohteen aspekteja (Runeson ja Höst 2009). Tämä työ voidaan siis jaotella toiminnallisen tutkimuksen alle, koska tarkoituksena on muokata ja kehittää Hiava OY:n toimintamallia ja luoda työskentelyä tehostava ohjelmistoratkaisu tutkimuksen pohjalta. (Yin 2009, s. 19) mukaan tapaustutkimuksessa on yleistä hyödyntää sekä määrällistä että laadullisia aineistoa, joka tekee tutkimusmenetelmästä monipuolisen ja joustavan. Tämän puolesta tapaustutkimukset eivät välttämättä sisällä suoraa ja yksityiskohtaista havainnointia, vaan nojaavat laadulliseen lähestymistapaan, kuten tämän työn tapauksessa voidaan tehdä käytettävyyden analysoinnissa.

Tapaustutkimuksissa voidaan, ja on suositeltavaa, käyttää useampia tiedonlähteitä. Näin voidaan paremmin välttyä siltä, ettei yksittäisen tiedonlähteen tulkinta luo liian yksipuoleista kuvaa. Johtopäätösten tekeminen on myös vahvempaa, jos useampi tiedonlähde osoittaa samanlaiseen suuntaan (Runeson ja Höst 2009). Tässä työssä hyödynnettävinä

tiedonlähteinä voidaan käyttää niin tarjolla olevaa kirjallisuutta kuin työn ohjaajaa sekä Hiava Oy:ltä saatuja ohjeita, kommentteja ja vaatimuksia.

Tässä työssä tapaustutkimuksen tukena käytettiin tutkimusmenetelmänä narratiivista kirjallisuuskatsausta. Pautassonin et al. (2019, s. 301) mukaan narratiivisen kirjallisuuskatsauksen tavoitteena on koota yhteen olemassa olevaa tietoa, jonka avulla voidaan välttää tarpeeton ja tietämättömyydestä johtuva aiemman tutkimuksen toistaminen. Tarpeettoman toiston välttämisen lisäksi tutkimuksen tarkoituksena on tarkastella ja hyödyntää kirjallisuudessa käsiteltäviä ohjelmistokehitykseen liittyviä ratkaisuja, joiden kautta voidaan luoda puitteet kehityksen tueksi.

3 Ohjelmistoratkaisun toteutuksesta

Ohjelmistokehitys kenttätyöhön ei ole pelkästään tekninen tehtävä, vaan se liittyy olennaisesti työn, tehtävien ja organisaation suunnitteluun. Rauterbergin, Strohmin ja Kirschin (1995) mukaan yksi ohjelmistokehityksen keskeisistä ongelmista on ollut kyvyttömyys tunnistaa ohjelmiston suunnittelu osana laajempaa työjärjestelmän kehittämistä. Tämän vuoksi ohjelmiston toteutuksessa tulisi alusta alkaen huomioida käyttäjien työtehtävät ja käytännön toimintaympäristö. Ohjelmiston toteutusvaiheessa keskeistä on suunnitella ratkaisu, joka on teknisesti toimiva, mutta ennen kaikkea käyttäjälähtöinen ja kenttätyöhön soveltuva. Tässä luvussa käsitellään ohjelmiston toteutukseen liittyviä keskeisiä osa-alueita, jotka pohjautuvat Hiava OY:n vaatimusmäärittelyihin ja tukevat kenttätyön sujuvuutta.

3.1 UI/UX suunnittelu

Arditon et al. (2014) mukaan yksi suurimpia ongelmia asiakkaan puolelta on käytettävyyden maininnan puute määrittely dokumentaatiossa, sekä ”kehittäjän ajattelumalli”, jonka mukaan kehittäjät keskittyvät mieluummin teknisiin haasteisiin ja teknisten toiminnallisuuksien toteuttamiseen kuin tuotteen käytettävyyteen. Tämän vuoksi tuotteen prototyypittäminen ja testaaminen loppukäyttäjällä on tärkeä osa kehitysprosessia. Testauksen aikana voi ilmetä ohjelmiston käyttöön ja käyttöliittymään liittyviä tekijöitä tai tarpeita, joita alkuperäisessä määrittelyssä ei ole otettu huomioon, tai joiden toteutus riippuu työympäristön luomista haasteista ja rajoitteista.

Työssä ohjelmiston UX testaamiseen sovelletaan siis prototyyppipohjaista lähestymistapaa, jossa alustavia ohjelmistoratkaisuja testataan käyttäjillä prosessin edetessä. Prototyyppitestauksessa luodaan matalan tarkkuuden esitys testattavasta asiasta (Coleman ja Goodwin 2017, s. 2). Tämä voidaan toteuttaa joko yksinkertaisella fyysisellä luonnoksella, jonka elementtejä voidaan liikutella ja vaihdella, tai interaktiivisella digitaalisella mallilla, jolla testataan tiettyä vuorovaikutusta tai toiminnallisuutta.

Käytettävyyden määritelmänä voidaan käyttää Kurosun (2015) esittämää käyttöastetta, jolla määritelty käyttäjä saavuttaa määritellyn tavoitteen määritellyssä tilanteessa tarkasti,

suorituskykyisesti ja tyydyttävästi. Tämän lisäksi käytettävyyttä voidaan mitata Weichbrothin (2024) mukaan oppimiskyvyllä, muistettavuudella, käytön aiheuttamalla kognitiivisella taakalla, virhesyötteillä, yksinkertaisuudella ja käyttömukavuudella. Esitetyistä määritteistä virhesyötteet ovat ainut selkeästi määrällinen mittari, ja loput ovat enemmän tai vähemmän laadullisia mittareita, joiden todentamiseen vaaditaan haastatteluja tai reaaliajassa tapahtuvaa käyttäjän seuraamista.

3.2 Datan käsittely ja siirtäminen

Työn esitietojen pohjalta datankeräys on pitkälti määrämuotoista ja vaatii vain pientä keräämisen jälkeistä siistimistä. Tämän oletuksen pohjalta tiedon jäsentelyyn ja tallentamiseen ei tarvita mitään erityisiä tietokantaratkaisuja tai tehokasta laitteistoa, vaan kerätty tieto voidaan jäsennellä ja lähettää analysointiin keräämiseen käytetyllä laitteella. Kerätyn datan tallentamisesta ei ole myöskään mitään mainintaa, joten oletuksena tieto tallennetaan paikallisesti keräyslaitteelle ja lähetetään siitä edelleen analysoitavan ohjelman käyttöön. Prosessointi ja datan lopullinen muoto tulee määräytymään Hiava Oy:n tarpeiden mukaan ja mahdollisten API (application programming interface)-vaatimusten pohjalta. API-vaatimuksilla tarkoitetaan tässä tapauksessa ohjelmiston sulauttamista osaksi Hiava Oy:n laajempaa ohjelmistokokonaisuutta ja ohjelmien välillä tapahtuvan viestinvaihdon standardisointia.

Ohjelmistojen välillä liikkuvan tiedon määrittelyn lisäksi on tärkeää määritellä, kuinka tiedon liikenne tapahtuu. Datan paikallisella prosessoinnilla voidaan vähentää mahdollisista verkkoyhteyden häiriöistä aiheutuvia ongelmia ja välttää vahingossa suojaamattomien lähiverkkojen kautta lähetettävän datan helppoa kaappausta (Issac ja Mohammed 2007). Jos data on arkaluontoista ja salassa pidettävää, olisi suositeltavaa siirtää se myös prosessoinnin jälkeen joko paikallisesti käyttökohteeseen tai langallista yhteyttä käyttäen. Isaacin ja Mohammedin (2007) mukaan langattomat verkot eivät ole koskaan täysin riski vapaita, mutta turvallisuutta voidaan parantaa erinäisin keinoin kuten vahvalla salauksella, vahvalla salasanalla, palomuurilla tai asettamalla laitekohtaisia suodattimia. Toisaalta langattoman verkon käyttö voi mahdollistaa datan välittömän analysoinnin päätepisteessä ja nopeamman päätöksenteon, joka voi olla tuon pienen turvallisuusriskin arvoista.

3.3 Digitaalinen datankeräys kenttätyössä

Käytettävyyden lisäksi työn toinen tärkeä osa-alue on itse datan keräämiseen käytettävä ohjelmisto. Riippuen asetetuista vaatimuksista, datan keräämiseen käytettävän ohjelmiston voi toteuttaa hyödyntämällä mahdollisia avoimen lähdekoodin vaihtoehtoja, tai luomalla yrityksen tarpeisiin räätälöidyn ratkaisun. Myös toteutukseen valitulla alustalla voi olla vaikutusta ratkaisumalliin. Mahdollisesti toteutettavina vaihtoehtoina voidaan pitää:

I. Android-järjestelmälle luotu ohjelmisto
II. yksinkertaisella käyttöliittymällä varustettu pc-ohjelmisto
III. komentorivin kautta ajettava ohjelmisto
IV. internet selaimen kautta täytettävä lomake

Järjestelmän ja lähestymistavan lopullinen valinta riippuu Hiava Oy:n tarpeista ja käyttäjien vaatimuksista ohjelmistolle. Isommalla ruudulla tai erillisillä hiirellä ja näppäimistöllä varustettu laite tarjoaa eri mahdollisuuksia pienempikokoiseen älypuhelimeen verrattuna.

3.3.1 Valmiit kaupalliset ratkaisut

Markkinoilta löytyy asiakaskyselyyn ja datankeräykseen useita valmiita kaupallisia vaihtoehtoja. Tässä työssä esimerkeiksi ja vertailuun on valittu yksinkertaisen internet haun pohjalta kolme helposti löydettävää ja näkyvää vaihtoehtoa, koska laajempi ja täysimääräisempi markkinatarjonnan analyysi menisi tämän työn määritysten ulkopuolelle. Tarkastelun kohteeksi on valittu FastField Forms (www.fastfieldforms.com), JotForm (www.jotform.com) ja SurveyCTO (www.surveycto.com). Alle on koottu yhteenveto merkittävimmistä yritysten mainostamista ominaisuuksista.

Taulukko 1. Kaupalliset vaihtoehdot

Ominaisuus	FastField Forms	JotForm	SurveyCTO
Mukautettavat lomakkeet	Kyllä	Kyllä	Kyllä
Offline-käyttö	Kyllä	Kyllä	Kyllä
Haarautumislogiikka	Kyllä	Kyllä	Kyllä
Raporttien generointi	Kyllä	Kyllä	Kyllä
Mediaominaisuudet (kuvat, GPS)	Kyllä	Kyllä	Kyllä
Datan laadunvalvonta	Ei	Ei	Kyllä
Alustat	iOS, Android, Web	iOS, Android, Web	iOS, Android, Web
Integraatio	API, 3. osapuolten integrointi	API, 3. osapuolten integrointi	API, 3. osapuolten integrointi
Tietoturva	Salaus	Salaus	Salaus ja käyttäjätasot
Hintataso	~22€ per käyttäjä per kuukausi	Rajoitettu 0-99€ per kuukausi	Alkaen ~200€/kk, lisähintaan lisää ominaisuuksia

Taulukon 1 perustella kaupalliset tuotteet ovat hyvin samankaltaisia, tarjolla samoilla alustoilla ja tarjoavat pitkälti samoja perusominaisuuksia. Yksi merkittävä ero on kuitenkin SurveyCTO:n tarjoama datan laadunvalvonta, jota muut vaihtoehdot eivät ominaisuuksissaan mainitse. Mikäli tämä ominaisuus on luotettavasti toteutettu, se voi olla ratkaiseva tekijä yrityksen datankeräyksessä, jossa virheellinen data voi johtaa virheellisiin ja toiminnan kannalta haitallisiin lopputuloksiin. Haarautumislogiikalla puolestaan tarkoitetaan toiminnallisuutta, jonka avulla tiettyihin kysymyksiin saatujen vastausten perusteella näytetään tai piilotetaan kyseiseen osioon liittyvät jatkokysymykset. Ominaisuudet kuten raporttien generointi tai GPS datan tallentaminen voivat olla yrityksen tarpeisiin turhia, jos kyseessä on vain datankeräys, ja sen myöhempi analysointi suoritetaan omilla ohjelmistoilla. Näiden ominaisuuksien tiedostaminen voi siitä huolimatta olla hyödyllistä, sillä ne voivat auttaa paremmin kartoittamaan omia tarpeita ja mahdollisuuksia.

Tuotteiden hinnoittelu riippuu JotFormin ja SurveyCT:n tapauksessa haluttujen ominaisuuksien määrästä, kun Fastfield puolestaan tarjoaa käyttäjäpohjaisella hinnoittelulla. JotForm on hinnoittelultaan alhaisin, mutta myös rajallisin. Ilmainen suunnitelma sallii 5

lomaketta, 100 kenttää ja 100 lähetystä kuukaudessa skaalautuen 99€ kuukautiseen 100 lomakkeen ja 10 000 lähetyksen suunnitelmaan. Kaikkien JotFormin suunnitelmien ehdoissa mainitaan yhden käyttäjän raja. Erillinen Enterprise-luokan suunnitelma rajattomilla ominaisuuksilla on erikseen neuvoteltavissa, eikä sille ole annettu suoraa hinnastoa. SurveyCTO:n tarjoama 200€ perus suunnitelmaa sallii asiakkaalle rajattoman määrän laitteita sekä käyttäjiä, 100 lomaketta ja 5 000 lähetystä. Suunnitelmat skaalautuvat ylös ~600€ kuukausilisenssiin, joka tarjoaa rajattomat ominaisuudet. Tämän lisäksi SurveyCTO tarjoaa myös Enterprise-luokan suunnitelmaa, joka on erikseen neuvoteltavissa.

3.3.2 Avoimen lähdekoodin esimerkki

Field Book (github.com/PhenoApps/Field-Book) on Android käyttöjärjestelmällä toimiva yksinkertainen avoimen lähdekoodin sovellus, joka luotiin kasvien fenotyyppisen tiedon keräykseen. AlMarzouqin et al. (2005) mukaan avoimen lähdekoodin ohjelmistot ovat vapaasti jaettavia, analysoitavia ja muokattavia ohjelmia, ja niiden käyttö osana laajempaa kokonaisuutta voi auttaa pienentämään kehityskuluja. Toisaalta ne voivat myös kasvattaa kuluja valmiisiin ratkaisuihin verrattuna, koska avoimen lähdekoodin osat tarvitsevat räätälöintiä ja yhdistelyä muihin osiin. Tämän vuoksi myös käyttöönotto voi olla kalliimpaa ja hitaampaa, koska käyttäjät tarvitsevat erityistä koulutusta.

Koska Field Book on kehitetty maatalous- ja kasvitutkimukseen, sen käyttöönotto muuhun tarkoitukseen tulee vaatimaan muokkausta ja räätälöintiä. Kyseinen ohjelmisto toimii tarvittaessa myös ilman verkkoyhteyttä, ja kerätty data tallennetaan paikallisesti laitteelle. Ohjelmistosta löytyy valmiina vastaavia ominaisuuksia kuin kaupallisistakin ohjelmistoista, kuten mukautettavat lomakkeet, kuvien ottaminen ja GPS tiedot. Ohjelmisto käyttää tiedon kommunikointiin kasvien lisääntymisdataan suunniteltua BrAPI (brapi.org) palvelua, jonka vuoksi ohjelmiston uuteen kontekstiin siirtäminen vaatii myös API palveluiden uudelleen järjestelyä tai mahdollista API skeeman lisäämistä.

4 Tapaus: Hiava Oy:n kenttädatankeräys

Tässä luvussa käsitellään Hiava Oy:n kenttätyöhön liittyviä tarpeita ohjelmistoratkaisulle. Luvussa käydään läpi ohjelmiston vaatimusmäärittely sekä teknisten ratkaisuiden valinta. Yrityksen tarpeisiin vaaditaan selainpohjainen ratkaisu, joka ohjaa käytettävien työkalujen ja teknologioiden valintaa. Vaatimusten pohjalta kartoitetaan mahdollisia teknisiä ratkaisuja ja pohditaan valmiiden kaupallisten sekä avoimen lähdekoodin ratkaisuiden soveltumista käytäntöön. Suunnittelun jälkeen valitaan totutustapa ja rakennetaan ohjelmistoratkaisun käyttöliittymä.

4.1 Vaatimusmäärittely

Projektin kirjallisuuskatsauksen pohjalta käytiin keskustelua Hiava Oy:n kanssa ohjelmistoratkaisun käytännön toteutuksesta ja tarpeista. Tavoitteena oli tunnistaa toiminnallisia ja teknisiä vaatimuksia, jotka tukevat kenttätyön sujuvuutta ja datankeräyksen luotettavuutta. Näiden pohjalta ohjelmistolle koottiin vaatimusmäärittely, joka on esitetty Taulukossa 2. Keskustelujen pohjalta pyrittiin luomaan tärkeysjärjestys vaatimuksille, jotta toteutus olisi järjestelmällisempää ja kattaa vähintään tärkeimmät osa-alueet. Ominaisuuksien arvottaminen on kehityksen kannalta oleellista, sillä se auttaa kohdentamaan resurssit oikein. Näin vältytään tilanteilta, joissa aikaa käytetään vähemmän kriittisten ominaisuuksien viimeistelyyn tärkeämpien toiminnallisuuksien kustannuksella. Selkeä priorisointi tukee myös vaiheittaista toteutusta ja mahdollistaa toimivan prototyypin rakentamisen rajallisessa ajassa

Taulukko 2: Ohjelmiston vaatimusmäärittely

ID	Vaatimus	Kuvaus	Tärkeys
0	Laitteistovaatimukset	Käyttö selaimessa, asiakkaat käyttävät omia päätelaitteitaan	Ehdoton
1	JSON-skeema	Lähtökohtaisesti oma skeema suunnitellaan käyttöön	Ehdoton
2	Datan muokkaus	Ohjelma muokkaa syötetyn datan tarkoitukseen	Ehdoton
3	Rooli: Asiantuntija	Organisaation asiantuntija tai Hiava Analyticsin valtuutettu henkilö, jolla on vastuu lomakerakenteista	Ehdoton
4	Tietotyypit	Ohjelmiston halutaan käsittelevän tekstiä, numerioita ja totuusarvoja	Ehdoton
5	Rooli: Asiakas	Henkilö, joka täyttää lomakkeen annetun rakenteen mukaan	Ehdoton
6	Käyttöliittymä	Osa Hiava Analyticsin UI:ta	Tärkeä
7	Integrointi	Yhdistys Analytics koodikantaan	Tärkeä
8	MVP-vaiheen erillisyys	Prototyyppi voi olla irrallinen ratkaisu kehityksen helpottamiseksi	Tärkeä
9	Kenttien yleiset rajoitukset	Kenttien muoto ja pakollisuus määritellään skeeman mukaan	Tärkeä
10	Numeroidan rajoitukset	Numeroiden arvovälit tulee määrittää skeeman mukaan	Tärkeä
11	Dynaamiset kentät	JSON-tietokenttien muokattavuus käyttäjän toimesta harkittava	Idea

Taulukon 2 vaatimusmäärittely on jaettu ehdottomiin, tärkeisiin ja idea tasolla oleviin ominaisuuksiin, joista ehdottomat toteutetaan ensimmäisenä ja tärkeät sekä ideat näiden jälkeen aikavaatimusten puutteissa. Toteutuksen vaatimuksena on luoda selainpohjainen ratkaisu, joka ohjaa käytettävien työkalujen valintaa. Vaatimusmäärittelyissä ei mainittu mitään erityistä teknisiin ratkaisuihin liittyen, joten valinta tehtiin vertailemalla mahdollisia vaihtoehtoja ja ottaen huomioon aiempi kokemus kehitysympäristöstä.

Koska projektin vaatimukset ovat suppeammat kuin markkinoilla olevien tuotteiden tarjonta, valmiin tuotteen käyttäminen luo päällekkäisyyksiä Hiavan käytössä olevan ohjelmiston kanssa ja taloudellisten resurssien tuhlaamista. Tämän lisäksi selainpohjainen ratkaisu vaatii integrointia osaksi Hiavan olemassa olevaa järjestelmää, jonka puolesta oman

selainpohjaisen ratkaisun kehittäminen nähtiin tarkoituksenmukaisempana kuin olemassa olevan avoimen lähdekoodin ohjelmiston sovittaminen. Avoimen lähdekoodin vaihtoehdot sisältävät paljon kontekstiin sopimattomia ominaisuuksia, eivät ole suoraan siirrettävissä selainkäyttöön, vaatisivat merkittävää räätälöintiä, API-rajapintojen uudelleenkirjoittamista sekä käyttöliittymän muokkausta vastaamaan Hiavan tarpeita. Tämä lisäisi kehitysaikaa ilman, että saavutetaan merkittävää etua suoraan vaatimuksiin perustuvan ja räätälöidyn ratkaisun rakentamiseen.

4.2 Teknisten ratkaisujen valinta

Toteutustavan valitsemisen jälkeen perehdyttiin moderniin web-kehitykseen ja saatavilla oleviin teknologioihin. Frontend-kehitykseen löytyi useita JavaScript pohjaisia vaihtoehtoja, mutta React-kirjasto valittiin aikaisemman kokemuksen, hyvän dokumentaation ja komponenttipohjaisen arkkitehtuurin vuoksi. Muita teknologioita, kuten laajoihin projekteihin skaalautuvaa Angularia (angular.dev) tai Vue.js:ää (vuejs.org) ei valittu, koska niiden käyttö olisi vaatinut oman lisäkoulutuksen ja perehdytyksen, eikä niiden käyttöönoton nähty tuovan mitään ilmeisiä etuja projektin tavoitteiden kannalta. Backend-testaukseen valittiin Node.js ja Express.js, koska ne tarjoavat nopean kehitysympäristön ja hyvän suorituskyvyn JSON-datan käsittelyyn. Valinta mahdollistaa myös saman ohjelmointikielen hyödyntämisen sekä front- että backendissä, mikä yksinkertaistaa kehitystä ja ylläpitoa.

4.2.1 Frontend-arkkitehtuurin suunnittelu

React:in valinta perustui sen komponenttipohjaiseen arkkitehtuuriin, joka tekee käyttöliittymän osien jakamisesta ja uudelleenkäytöstä selkeää. Käyttöliittymäkomponenteissa hyödynnettiin Bootstrap-kirjastoa, joka kautta projektiin voidaan tuoda valmiita responsiivisia elementtejä ja nopeuttaa kehitystä ilman, että kaikkia elementtejä tarvitsee luoda ja muokata käsin projektin edetessä. Näiden ominaisuuksien avulla on tarkoitus nopeuttaa käyttöliittymän kehitystä ja lyhentää testaamiseen tarvittavaa aikaa.

Toteutettava käyttöliittymä on tarkoitus integroidan osaksi Hiava Analyticsin kokonaisratkaisua, eivätkä Bootstrap-kirjaston kautta tuodut oletuskomponentit ole välttämättä lopullinen ratkaisu. Valmiiden komponenttien lisäksi React mahdollistaa myös omien CSS-tyylien käytön luomalla halutulle komponentille uuden luokan ja asettamalla kyseinen luokka osaksi komponentin määrittelyä, joten valmiiden komponenttien hyödyntäminen prototyypissä on projektin kannalta perusteltu valinta.

Käyttöliittymä jaettiin kolmeen pääkomponenttiin: lomakkeiden rakentajaan (FormBuilder), lomakkeiden täyttäjään (FormFiller) ja vahvistusnäkymään (ConfirmSubmission). Tämä jako selkeyttää vastuunjakoa ja helpottaa koodin ylläpitoa. Testaamiseen luotiin myös erillinen käyttäjän kirjautumista jäljittelevä aloitussivu, jonka kautta asiakaskäyttäjän ja ylläpitäjän oikeuksista vastaavia komponentteja voitiin testata. Navigaatiokomponentti (App.jsx) puolestaan vastaa sivuston sisäisestä reitityksestä sekä yhteisistä käyttöliittymäelementeistä.

4.2.2 Backend testiarkkitehtuurin suunnittelu

Express.js-palvelin valittiin sen yksinkertaisuuden ja tehokkuuden vuoksi RESTful API:n toteuttamiseen. Projektissa käytettiin Vite-kehitystyökalua, joka mahdollistaa nopeamman kehitysympäristön ja ominaisuuksien testaamisen sekä muokkauksen selaimen nopean uudelleenlatauksen ansiosta. CORS-konfiguraatio lisättiin mahdollistamaan frontend- ja backend-komponenttien välisen kommunikaation eri porteilla kehitysvaiheessa.

Kehitysvaiheessa käytettiin mock-dataa, mikä nopeuttaa kehitystä ja mahdollistaa nopean testauksen. Tämä ratkaisu on sopiva prototyypin kehitykseen, mutta tuotantoversiota varten tarvitaan laajempi tietokantarakenne. Backend ratkaisua käytettiin mockup-palvelimen pyörittämiseen, jolloin projektin testaaminen voidaan toteuttaa suljetussa ja turvallisessa ympäristössä. Käyttäjien pääsy tietokantaan rakennettiin käyttäjälle luodun kyselylistan pohjalta. Jos kyseessä on asiakaskäyttäjä (customer), hakee järjestelmä käyttäjälle tallennetun listan käyttäjänimen perusteella. Asiantuntijan (expert) tapauksessa palvelin palauttaa kaikkien tietokannasta löytyvien kyselyiden nimet. Haetun listan perusteella sivusto esittää käyttäjälle asetettujen lomakkeiden nimen, jonka kautta käyttäjä voi hakea täytettävän lomakkeen tiedot palvelimelta.

Käyttäjän autentikointi on toteutettu JWT-pohjaisesti, jossa käyttäjän roolipohjainen valvonta antaa pääsyn ominaisuuksiin. Prototyyppiversiosta löytyy ”customer” ja ”expert” käyttäjät, joista ”customer” voi täyttää lomakkeita ja ”expert” täyttää sekä luoda uusia lomakkeita. Kovakoodatut käyttäjätiedot ja tokenin tallennus selaimen localStorage-muistiin simuloivat tuotantoympäristöä ja luovat testausvaiheessa nopeasti toteutettavat ja selkeät raamit käyttäjänhallintaan. Palvelimen puolella toteutettiin `/login`-loppupiste, joka vastaanottaa käyttäjänimen ja salasanan, tarkistaa ne mock-tietokannasta ja palauttaa JWT-tokenin onnistuneen kirjautumisen yhteydessä. Tokkeni tallennetaan paikallisesti käyttäjälle ja lähetetään palvelimelle varmennettavaksi toimintojen yhteydessä.

4.2.3 Roolipohjainen käyttäjäoikeus

Prototyypin roolipohjainen pääsy ominaisuuksiin toteutettiin Reactin koukku (Hook) toiminnon avulla, joka mahdollistaa tietojen tallentamisen istunnon ajaksi. Koukkujen kautta ohjelma tarkastaa käyttäjän kirjaaman roolin serveriltä `/protected`-loppupisteen kautta ja evää pääsyn sivuille, joihin käyttäjällä ei ole oikeutta. Tämä ratkaisu mahdollistaa helpon roolien lisäämisen tulevaisuudessa, toimintojen muokkaamisen tuotantoversioon ja yhdistämisen Hiavan omaan käyttäjänhallintajärjestelmään. Käyttäjätiedot lähetetään jokaisen palvelinpyynnön yhteydessä ja voidaan varmentaa palvelimen toimesta, joka parantaa tietoturvaa ja kieltää tietoihin pääsyn luvattomilta käyttäjiltä.

4.3 Käyttöliittymän toteutus

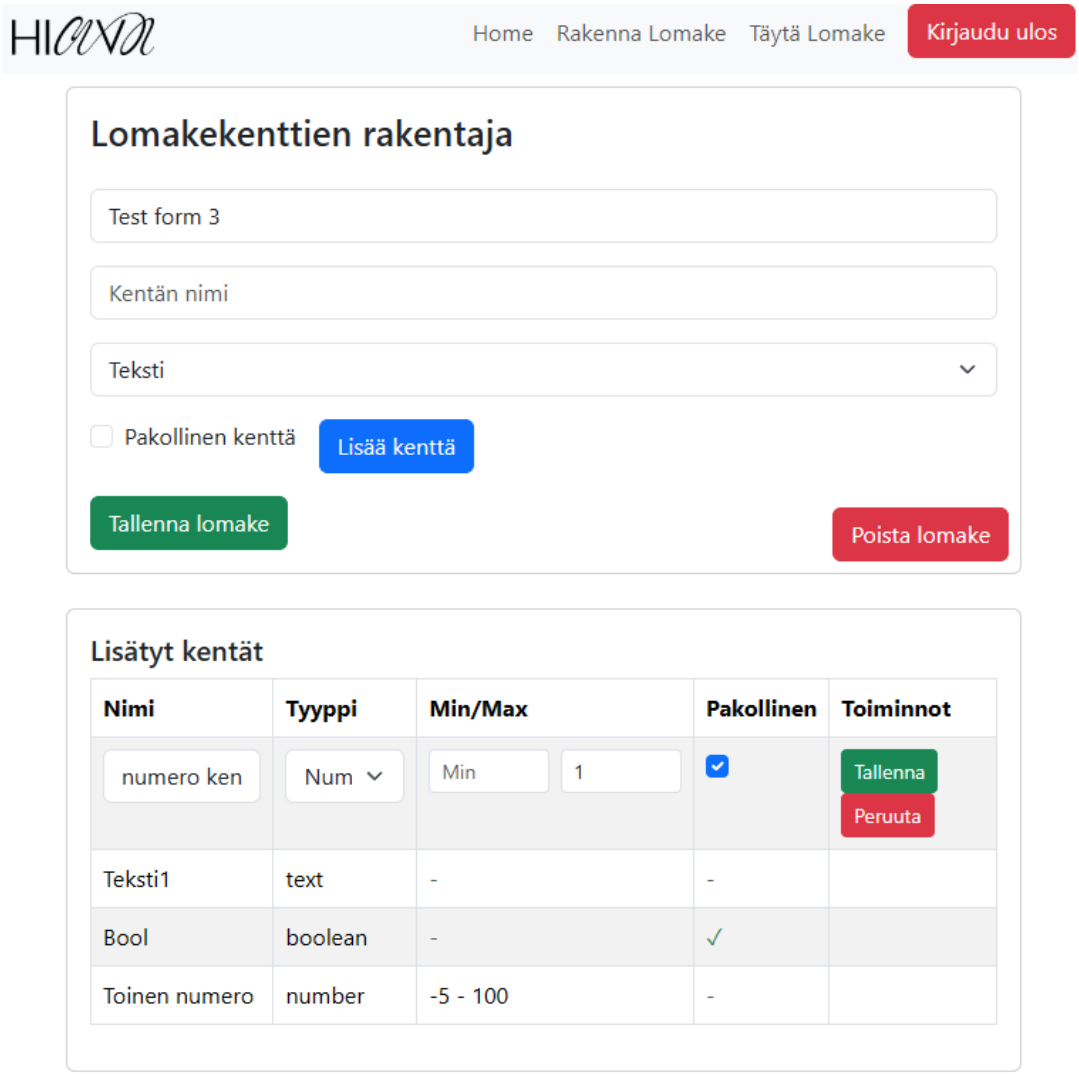
Käyttöliittymän navigointiin käytettiin React Router-kirjastoa. Sivun ylälaidassa sijaitseva navigaatiopalkki sisältää linkit prototyypin eri sivuille. Navigaatiopalkki on kiinnitetty sivun yläosaan, ja se on kaikille yhteinen elementti, joka mahdollistaa navigoinnin kaikilta sivuilta. Palkin luomiseen käytettiin Bootstrap-kirjaston navbar-komponenttia, joka mahdollistaa responsiivisen suunnittelun ja palkin mukautumisen eri kokoisille näyttöalustoille. Etenkin käyttäjänhallintaan liittyvät näppäimet löytyvät monilta sivustoilta, kuten Googlen eri palvelut, Reddit, Microsoftin palvelut, Nytimes jne., sijoitettuna sivuston oikeaan yläkulmaan. Tämän lisäksi navigaatiopalkki on usein sivun ylälaidassa, joka tuntui etenkin kapeammilla näytöillä käyttäjäystävällisemmältä kuin sivulla oleva palkki.

Käyttöliittymän komponentit käyttävät yleisesti Bootstrap:in grid-järjestelmää ja luokkia, mikä mahdollistaa yhdenmukaisen ulkoasun ei laitteilla. Tiedonsyötöstä vastaavat komponentit puolestaan rakennettiin korttien ja taulukoiden pohjalle, joka mahdollistaa dynaamisen objektien lisäämisen ja selkeän organisoinnin. Objektien syötekenttiin on rakennettu varmistukset, jotta syötettävä tieto on varmasti oikeaa tyyppiä. Numeroita kysyviin kenttiin voidaan syöttää vain numeroita. Tekstikentille ei ole erikseen määrityksiä syötteen pituudelle, vaan tämä tarkastetaan kaikkia kenttiin koskevalla maksimi merkkien määrän ”maxLength” kautta. Tekstiä kysyviin kenttiin voidaan syöttää kaikkia merkkejä, jonka lisäksi tekstistä leikataan pois aloittavat ja lopettavat välilyönnit. Tällä vältetään pelkkiä välilyöntejä sisältävien vastausten hyväksyminen.

Numeroiksi määritellyille kentille on määriteltävissä minimi ja maksimi arvot, joista toinen tai molemmat voidaan jättää myös määrittelemättä. Uutta lomaketta luodessa arvot annetaan kenttäkohtaisesti. Lomaketta täyttäessä nämä arvot asetetaan palvelimelta haetun lomakkeen pohjalta lomakkeen kentille. Tämän lisäksi kentille voidaan valita ”mandatory” vaihtoehto, joka tekee niistä joko pakollisia tai valinnaisesti täytettäviä. Prototyypin väri ja tyyli perustuu Bootstrapin oletusteemoihin. Teema tullaan myöhemmin muokkaamaan Haivan brändin mukaiseksi mahdollisesti osana integraatiota, koska teeman värimaailmalle ja muodoille ei annettu ohjeita vaatimusmäärittelyissä. Tästä syystä kehityksen kannalta järkevämmäksi koettiin olla käyttämättä kehitysaikaa ulkoasun teemaan ja keskittyä ulkoasuun vain toiminnallisuuden ja käyttäjäkokemuksen näkökulmasta.

4.3.1 Dynaaminen lomakkeiden luonti

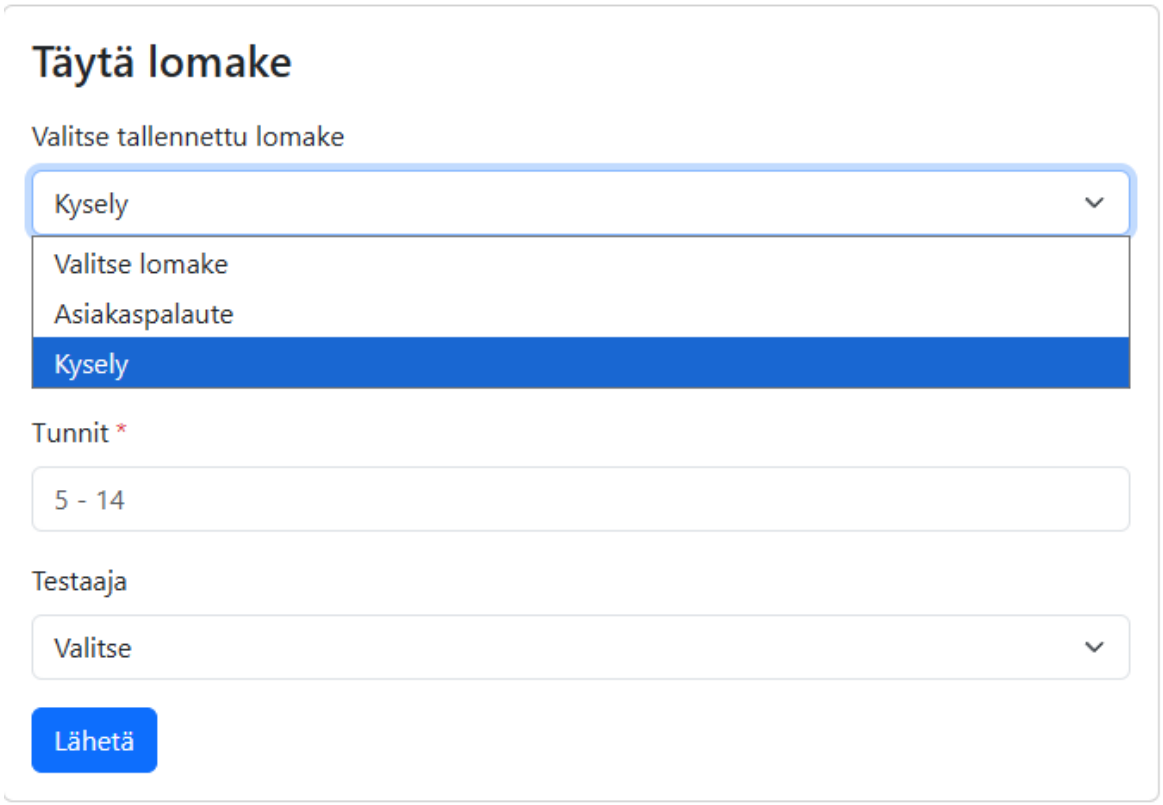
Uusien lomakkeiden rakentaminen toimii FormBuilder-komponentin kautta. Komponentti mahdollistaa dynaamisen lomakkeiden luonnin Reactin tilanhallinnan kautta, joka mahdollistaa sivuston reaaliaikaisen päivityksen käyttöliittymässä. Käyttäjä voi määritellä lisättävälle lomakkeen kentälle nimen ja valita kenttään syötettävän arvon tyypin. Kenttien mahdolliset tyypit ovat teksti (string), numero (number) ja totuusarvo (boolean). Lisätyt kentät lähetetään palvelimelle JSON muotoisena tiedostona, joka sisältää lomakkeen nimen, kenttien nimet sekä niiden tyypit.



Kuva 1. Lomakekenttien rakentaja

Kenttää luodessa käyttäjä voi kuvan 1 mukaisesti nimetä lomakkeen sekä lisättävät kentät. Kenttien tiedot ovat myös muokattavissa lisäämisen jälkeen, sekä kenttiä voi poistaa halutessaan. Testauksen aikana koettiin hyödylliseksi myös lisätä ominaisuus, joka piilottaa muiden kenttien toimintonäppäimet, kun yhden kentän muokkaaminen on käynnissä. Käyttäjän tallentaessa lomaketta avautuu sivulle ”alert” tyyppinen varmistusviesti, joka kysyy käyttäjältä varmistuksen lomakkeen tallentamisesta. Tämän lisäksi lomakkeeseen lisättiin virheilmoituksia kertomaan puuttuvista kenttien nimistä, identtisistä kenttien nimistä ja syötteiden pituudesta. Lomakkeiden poistaminen on toteutettu ”Poista lomake” näppäimen kautta, joka avaa varmistus elementin ja lähettää hyväksynnän jälkeen lomakkeen nimikenttään syötetyn tekstin palvelimelle postamista varten.

4.3.2 Lomakkeiden täyttäminen



Kuva 2. Lomakkeen täyttäminen

Käyttäjän siirtyessä lomakkeiden esittämisestä vastaavalle sivulle lähetetään palvelimelle hakupyyntö käyttäjälle saatavilla olevista lomakkeista. Käyttäjä voi kuvan 2 mukaisesti valita pudotusvalikosta lomakkeen nimen, jonka haluaa ladattavaksi näytölle. Lomakkeen tiedot haetaan listan nimen mukaan palvelimelta ja esitetään sivulla. Lomakkeelle määriteltyjen kenttien pohjalta käyttäjä voi joko kirjoittaa tekstiä, kirjoittaa numeroita tai valita pudotusvalikosta totuusarvon riippuen mikä tietotyyppi kentälle on lomaketta luodessa asetettu. Jos lomakkeen kentälle on asetettu arvovälit, ne esitetään tyhjässä kentässä. Pakolliseksi määritetyt kentät on ilmaistu punaisella ”*” kentän nimen perässä.

Kun lomakkeen pakolliset kentät on täytetty, voi käyttäjä siirtyä ”Lähetä” painikkeesta varmistamaan tiedot, palaamaan takaisin, jos jotain on täytetty väärin, tai lähettää tarkastetun lomakkeen palvelimelle. Lähetyksen aikana ”Lähetä” näppäin otetaan pois käytöstä, jotta vältytään vahingossa lähettämästä lomaketta useampaan kertaan. Lähetyksen jälkeen käyttäjälle esitetään viesti joko onnistuneesta lähetyksestä tai lähetyksen epäonnistumisesta, jotta käyttäjälle ei jää epäselvyyttä toiminnon käytöksestä.

5 Keskustelu

Ohjelmistoratkaisuun saatiin toteutettua välttämättömät määrittelyt, mutta tarkkojen UI- ja rajapintamäärittelyiden puutteen vuoksi tietokantaan integrointi ja Hiava Analytics UI:n mukainen ulkoasu jäivät toteuttamatta. Toteutettu ratkaisu vastasi asetettuja tavoitteita: se mahdollistaa dynaamisten lomakkeiden luonnin, sisältää roolipohjaisen käyttäjähallinnan ja tarjoaa perustoiminnot kenttäoperaattoreiden toistuvien tehtävien tueksi. Tutkimuksen aikana onnistuttiin osoittamaan, että digitaalinen tiedonkeruu voi parantaa tehokkuutta, kerättävän tiedon tarkkuutta ja käytettävyyttä verrattuna fyysisiin tiedonkeräyksen menetelmiin. Digitalisointi antaa myös potentiaalisia hyötyjä vasteajoissa, jos kerätty tieto saadaan välittömästi jatkoprosessoitavaksi.

React ja Bootstrap tarjosivat hyvät lähtökohdat kehitykselle ja osoittautuivat tehokkaaksi sekä nopeaksi työkaluksi ulkoasun iteroimiselle. Hyvä dokumentointi auttoi myös huomattavasti kehityksessä hyödynnettyjen ominaisuuksien toteuttamisessa ja ongelmanratkaisussa. Prototyyppipohjainen UX-testaus osoittautui toimivaksi lähestymistavaksi ja auttoi tunnistamaan käytettävyyteen liittyviä haasteita jo varhaisessa kehitysvaiheessa.

Vaikka toteutettu järjestelmä vastaa sille asetettuja perusvaatimuksia, tutkimukseen liittyi myös rajoitteita. Käytettävyystestaus toteutettiin hyvin rajallisella käyttäjäjoukolla, mikä rajoittaa tulosten yleistettävyyttä. Myös tietoturvan tarkempi analysointi jäi alustavalle tasolle, koska kehitysvaiheessa ei ollut pääsyä Hiava OY:n omaan käyttäjätietokantaan tai tietoturvan implementaatioon. Ohjelmiston laajempi suorituskyvyn ja optimoinnin arviointi jätettiin myös tutkimuksessa rajauksen ulkopuolelle. Ohjelmiston kehitys tapahtui erillisessä kehitysympäristössä, eikä projektia testattu suoraan Hiava OY:n verkkosivujen yhteydessä. Tämän vuoksi esimerkiksi suurten datamäärien käsittelyyn ja prototyypin skaalautuvuuteen liittyviä ongelmia ja haasteita ei kyetty suuremmin tarkastelemaan. Täysi integrointi tai osittainen yhdistäminen osaksi verkkosivuja tulee vaatimaan myös lisää testausta ja muutoksia, koska käytännön toimivuutta tuotantoympäristössä ja sovittamista olemassa olevaan tietokantaan ei ole voitu arvioida.

Työn tulosten perusteella voidaankin esittää useita jatkokehityksen suuntia. Ohjelmiston suorituskykyä ja tietoturvaa olisi hyvä tarkastella syvällisemmin erityisesti tilanteissa, joissa

datan ja käyttäjien määrä kasvaa tai käyttöympäristö monimutkaistuu. Integraatio osaksi olemassa olevaa järjestelmää on toinen ilmeinen kohde jatkokehitykselle. Järjestelmässä ei myöskään ole tällä hetkellä ominaisuutta lisättyjen lomakkeiden muokkaamiseen, joka voisi olla käytön kannalta hyödyllistä. Koodin selkeyden kannalta voisi olla hyödyllistä siirrellä jotain funktioita kuten palvelimen kanssa kommunikointia omaan tiedostoonsa. Käytettävyystestausta olisi hyvä laajentaa suurempaan ja monipuolisempaan käyttäjäryhmään esimerkiksi hyödyntämällä gainsharing-mallia, jonka kautta järjestelmässä olevat positiiviset ja negatiiviset puolet saadaan luotettavammin esiin.

Työssä painotetiin ohjelmiston perustoimintojen toteutusta, käyttöliittymän selkeyttä ja luotettavuutta itse käyttöliittymän ulkonäön jäädessä vähemmälle huomiolle. Tavoitteena oli luoda selkeä ja helppokäyttöinen käyttöliittymä, joka ei vaadi laajempaa tietoteknistä osaamista. Tärkeää oli myös luoda järjestelmä, jossa virheiden tekeminen on vaikeaa, virheistä palautuminen on helppoa, eikä virheiden korjaaminen aiheuta käyttäjälle turhautumista ja altista uusille virheille.

Ohjelmistoratkaisu toteutettiin yksittäisenä tapaustutkimuksena, eikä tulosten perusteella voi yleistää päätelmiä koskemaan yleisesti kenttädatan keräämistä koskeviin tarpeisiin. Työ antaa kuitenkin viitteitä siihen, että asiakasdatan keräämisen digitalisointi on hyvin toteutettavissa eri käyttäjäryhmien tarpeisiin. Yrityksen olemassa oleviin järjestelmiin integroinnin kannalta voi olla hyödyllistä selvittää ja esittää ohjelmistojen tarkemmat rajapinnat, jotta kehityksen kannalta on helpompi huomioida yrityksen tarpeet, vältytään rajapintojen kehityksiin liittyviltä epävarmuuksilta ja mahdollisesti nopeutetaan lopullista käyttöönottoa.

6 Yhteenveto

Tässä työssä tutkittiin selainpohjaisen prototyypin kehittämistä Hiava Oy:n kenttäoperaattoreiden toistuvien asiakasdatan keräystarpeiden digitalisoimiseksi. Työn lähtökohtana oli parantaa tiedonkeräysprosessin tehokkuutta, tarkkuutta ja käytettävyyttä luomalla ohjelmistoratkaisu, joka on integroitavissa yrityksen olemassa oleviin järjestelmiin. Tutkimus toteutettiin tapaustutkimuksena ja narratiivisena kirjallisuuskatsauksena, joiden avulla selvitettiin paperipohjaisen ja digitaalisen tiedonkeräyksen keskeisiä haasteita sekä käytettävyyteen liittyviä kriteereitä. Kirjallisuudesta tunnistettiin neljä käytettävyyden pääattribuuttia: tarkkuus, suorituskyky, tyytyväisyys ja oppimiskyky. Tulokset viittasivat siihen, että digitalisoinnilla voidaan tehostaa kenttätyötä, parantaa datan laatua ja hyödyntää tehokkaammin käytettävissä olevia resursseja.

Vaatimusmäärittelyissä korostuivat selainkäyttö, JSON-skeeman pohjautuva dynaaminen lomakerakenne ja roolipohjainen käyttäjähallinta. Tämän lisäksi tehtiin suppea vertailu kaupallisten ratkaisuiden ominaisuuksista ja avoimen lähdekoodin ratkaisuista. Vertailun pohjalta todettiin räätälöidyn selainpohjaisen ratkaisun vastaavan parhaiten Hiava Oy:n tarpeita, joka rakennettiin React- ja Bootstrap-teknologioilla (frontend) sekä Node.js- ja Express.js-ympäristössä (backend). Prototyyppipohjainen UX-testaus vaikutti toimivalta ratkaisulta ja auttoi tunnistamaan käytettävyysongelmia varhaisessa vaiheessa. Rajoitteina olivat rajallinen käyttäjätestaus, laajemman tietoturva-analyysi puute ja suljettu kehitysympäristö.

Jatkokehityksessä suositellaan laajempaa käytettävyystestausta, tietoturvan ja suorituskyvyn syvällisempää arviointia, mahdollisia ominaisuuksien lisäämistä sekä integraatiota ja ulkoasun yhdentämistä Hiava Oy:n järjestelmiin. Työ tarjoaa käytännön esimerkin paperisen datankeräyksen siirtämisestä digitaaliseen muotoon ja toistuvien .

Lähteet

AlMarzouq, M. ym. (2005) ”Open Source: Concepts, Benefits, and Challenges”, Communications of the Association for Information Systems, 16. Saatavissa: https://doi.org/10.17705/1CAIS.01637.

Ardito, C. ym. (2014) ”Investigating and promoting UX practice in industry: An experimental study”, International Journal of Human-Computer Studies, 72(6), s. 542–551. Saatavissa: https://doi.org/10.1016/j.ijhcs.2013.10.004.

Boehm, B.W. ja Papaccio, P.N. (1988) ”Understanding and controlling software costs”, IEEE Transactions on Software Engineering, 14(10), s. 1462–1477. Saatavissa: https://doi.org/10.1109/32.6191.

Burgdorf, K.S. ym. (2016) ”Digital questionnaire platform in the Danish Blood Donor Study”, Computer Methods and Programs in Biomedicine, 135, s. 101–104. Saatavissa: https://doi.org/10.1016/j.cmpb.2016.07.023.

Coleman, B. ja Goodwin, D. (2017) Designing UX: prototyping. Collingwood, VIC Australia: SitePoint Pty. Ltd. (O’Reilly) : SitePoint.

Fletcher, L.A. ym. (2003) ”Handheld Computers: A Feasible Alternative to Paper Forms for Field Data Collection”, Evaluation Review, 27(2), s. 165–178. Saatavissa: https://doi.org/10.1177/0193841X02250527.

Happonen, A. ja Ghoreishi, M. (2022) ”A Mapping Study of the Current Literature on Digitalization and Industry 4.0 Technologies Utilization for Sustainability and Circular Economy in Textile Industries”, teoksessa X.-S. Yang ym. (toim.) Proceedings of Sixth International Congress on Information and Communication Technology. Singapore: Springer Singapore (Lecture Notes in Networks and Systems), s. 697–711. Saatavissa: https://doi.org/10.1007/978-981-16-2102-4_63.

Happonen, A. ja Siljander, V. (2020) ”Gainsharing in logistics outsourcing: trust leads to success in the digital era”, International Journal of Collaborative Enterprise, 6(2), s. 150. Saatavissa: https://doi.org/10.1504/IJCENT.2020.110221.

Issac, B. ja Mohammed, L.A. (2007) ”War Driving and WLAN Security Issues—Attacks, Security Design and Remedies”, Information Systems Management, 24(4), s. 289–298. Saatavissa: https://doi.org/10.1080/10580530701585831.

Kortelainen, H., Happonen, A. ja Kinnunen, S.-K. (2016) ”Fleet Service Generation—Challenges in Corporate Asset Management”, teoksessa K.T. Koskinen ym. (toim.) Proceedings of the 10th World Congress on Engineering Asset Management (WCEAM 2015). Cham: Springer International Publishing (Lecture Notes in Mechanical Engineering), s. 373–380. Saatavissa: https://doi.org/10.1007/978-3-319-27064-7_35.

Kortum, P. ja Sorber, M. (2015) ”Measuring the Usability of Mobile Applications for Phones and Tablets”, International Journal of Human-Computer Interaction, 31(8), s. 518–529. Saatavissa: https://doi.org/10.1080/10447318.2015.1064658.

Kurosu, M. (2015) Human-Computer Interaction: 17th International Conference, HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015. Proceedings, Part I. Cham: Springer International Publishing AG (Lecture Notes in Computer Science Ser, v. 9169).

Ozok, A.A. ym. (2008) ”A Comparative Study Between Tablet and Laptop PCs: User Satisfaction and Preferences”, International Journal of Human-Computer Interaction, 24(3), s. 329–352. Saatavissa: https://doi.org/10.1080/10447310801920524.

Pautasso, M. (2019) ”The Structure and Conduct of a Narrative Literature Review”, teoksessa M. Shoja ym. (toim.) A Guide to the Scientific Career. 1. p. Wiley, s. 299–310. Saatavissa: https://doi.org/10.1002/9781118907283.ch31.

Rauterberg, M., Strohm, O. ja Kirsch, C. (1995) ”Benefits of user-oriented software development based on an iterative cyclic process model for simultaneous engineering”, International Journal of Industrial Ergonomics, 16(4–6), s. 391–409. Saatavissa: https://doi.org/10.1016/0169-8141(95)00021-8.

Runeson, P. ja Höst, M. (2009) ”Guidelines for conducting and reporting case study research in software engineering”, Empirical Software Engineering, 14(2), s. 131–164. Saatavissa: https://doi.org/10.1007/s10664-008-9102-8.

Rymarczyk, J. (2020) ”Technologies, Opportunities and Challenges of the Industrial Revolution 4.0: Theoretical Considerations”, Entrepreneurial Business and Economics Review, 8(1), s. 185–198. Saatavissa: https://doi.org/10.15678/EBER.2020.080110.

Weichbroth, P. (2024) ”Usability of Mobile Applications: A Consolidated Model”, IEEE Access, 12, s. 62378–62391. Saatavissa: https://doi.org/10.1109/ACCESS.2024.3395528.

Yin, R.K. (20) Case study research: design and methods. 4. ed., [Nachdr.]. Los Angeles, Calif.: Sage (Applied social research methods series, 5).