TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Sander Põldma

„Kasutatavuse kogemuspõhise analüüsi meetodi“ rakenduse edasiarendus ja tiimitöö juhtimine

Bakalaureusetöö (9 EAP)

Juhendaja:
Anu Piirisild, MSc

Tartu 2025

„Kasutatavuse kogemuspõhise analüüsi meetodi“ rakenduse edasiarendus ja tiimitöö juhtimine

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärk oli viia läbi EbA-UIM rakenduse arendamise teine etapp, keskendudes nii tehnilistele aspektidele kui ka tiimitöö juhtimisele. Töö raames täiendas autor rakenduse tagasüsteemi funktsionaalsusi, optimeeris eessüsteemi andmetabelite kuvamist ning muutis rakenduse Electroni raamistiku abil platvormist sõltumatuks. Projekt viidi läbi meeskonnatööna, kus töö autor juhtis projektihaldust ja koordineeris tiimitööd, kasutades selleks projektihaldustarkvara ning kombineerides traditsioonilisi ja agiilseid arendusmetoodikaid. Töö tulemusena valmis täiustatud ja kasutajasõbralikum vabavaraline EbA-UIM rakendus.

Võtmesõnad: EbA meetod, Bitbucket, Jira, Electron, Apache POI, Jackson, projektijuhtimine, koskmudel, agiilne arendusmetoodika, hübriidmudel

CERCS: P175 Informaatika, süsteemiteooria

Further Development and Team Leadership for the „Experience-Based Analysis Method“ Application

Abstract:

The aim of this bachelor's thesis was to carry out the second phase of the development of the EbA-UIM application, focusing on both technical aspects and team management. During the project, the author further developed the application's backend, optimized the frontend table display, and made the application cross-platform using the Electron framework. The project was conducted as a team effort, with the author leading project management and coordinating team activities by using project management software and combining traditional and agile development methodologies. As a result, an improved and more user-friendly open-source version of the EbA-UIM application was completed.

Keywords: EbA method, Bitbucket, Jira, Electron, Apache POI, Jackson, team lead, Waterfall, Agile development, Hybrid model

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Kasutatavuse kogemuspõhise analüüsi (ingl *Experience-based Analysis*, EbA) meetod on Anu Piirisilla loodud kasutatavuse inspekteerimise meetod, mida kasutatakse digitoote või e-teenuse nõuete analüüsi faasis. EbA meetod aitab kasutajate varasemaid käitumisharjumusi analüüsides mõista, kas planeeritud lahendus hakkab toimima või vajab muutmist [1]. EbA tulemused aitavad leida nõuetes kohad, mida on vaja muuta või täiendada, muutes seeläbi kogu arenduse protsessi kiiremaks ja rohkem kuluefektiivsemaks [1].

EbA meetodi rakendamiseks on varasemalt loodud töövahend nimega EbA-UIM, mis valmis arenduse esimeses etapis Iris Kreinini [2] ja Karl Olaf Kuldmaa [3] koostööna. Esimeses etapis loodud rakenduses on võimalik läbi viia analüüs vastavalt EbA töövahendi kasutamise protsessile [2]. Tooteomanik soovis jätkata teise etapi arendustega, et rakendust veelgi kasutajasõbralikumaks muuta.

Siinse bakalaureusetöö eesmärgiks on teostada EbA-UIM rakenduse arendamise teises etapis järgmised ülesanded, mille võib jagada neljaks suuremaks osaks. Esiteks ees- ja tagasüsteemi edasi arendamine. Teiseks EbA-UIM rakenduse platvormist sõltumatuks muutmine ja installeerija loomine. Kolmandaks rakenduse optimeerimine, et tagada tänapäevastele standarditele vastav töökiirus ka suuremate andmemahtude juures. Neljandaks tiimitöö juhtimine ja projektihaldus, mis hõlmab endas efektiivse töökeskkonna ülesseadmist, projektijuhtimismetoodika valimist, regulaarsete koosolekute läbiviimist ja tiimiliikmetele tehnilise abi pakkumist. EbA-UIM rakenduse arenduse teine etapp teostatakse tiimitööna. Lisaks siinse bakalaureusetöö autorile kuuluvad tiimi Gerdo Germann, Rahel Pettai ja Timo Kaasik. Tellija ehk tooteomaniku positsiooni täidab Anu Piirisild.

Käesolev bakalaureusetöö on jaotatud kolmeks suuremaks peatükiks. Esimeses peatükis avatakse EbA meetodi olemus, tutvustatakse EbA-UIM rakenduse arenduse esimest etappi ning tuuakse välja teise etapi eesmärgid ja tiimiliikmete tööülesanded. Samuti tutvustatakse tarkvaraprojekti ülevõtmisel tähelepanu vajavaid aspekte, erinevaid platvormist sõltumatuid raamistikke, projektihaldustarkvarasid ja projektijuhtimise metoodikaid. Töö teine peatükk tutvustab täpsemalt EbA-UIM rakenduse teise etapi arendusprotsessi: ees- ja tagasüsteemi edasiarendust, installeerimisloogika muutmist ja tiimitöö juhtimist. Kolmandas peatükis tuuakse välja tulemused kasutajatega testimiselt ning arutletakse töös tehtud valikute üle. Töös kasutatakse tarkvaraarenduses kasutatavate terminite tõlkimiseks Cybernetica andmekaitse ja infoturbe portaali [4] või Heikki Vallaste e-teatmikku [5].

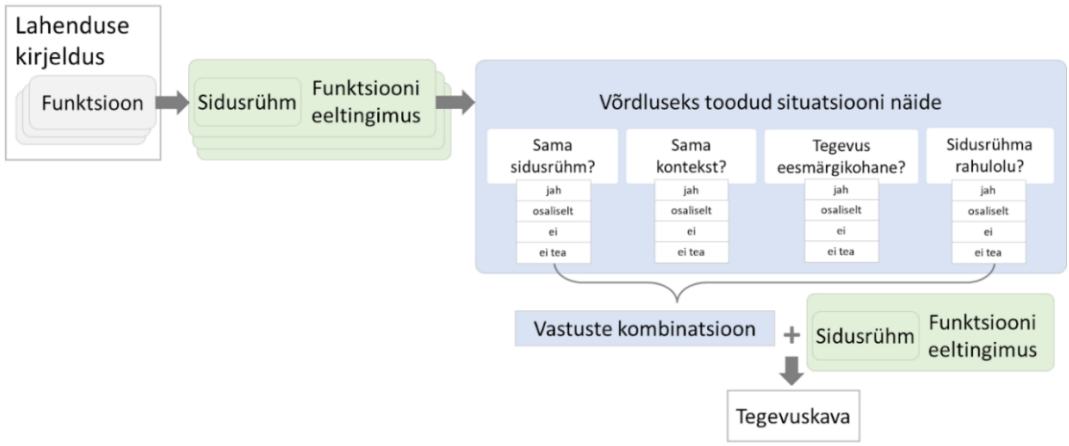
1. Taust

Käesolevas peatükis kirjeldatakse EbA meetodit ja sellele loodud töövahendit, võetakse kokku EbA-UIM rakenduse arenduse esimene etapp ning kirjeldatakse lühidalt teise etapi eesmärke, antakse ka ülevaade teise etapi tiimist ja tiimiliikmete ülesannetest ning käesoleva bakalaureusetöö eesmärgist. Peatükis tutvustatakse tarkvaraprojekti ülevõtmisel tähelepanu vajavaid aspekte, erinevaid platvormist sõltumatuid raamistikke, projektihaldustarkvarasid ja projektijuhtimise metoodikaid.

1.1 EbA meetodi ja töövahendi kirjeldus

EbA meetodit tutvustav peatükk tugineb Anu Piirisilla jt artiklile [1], kasutatavuse kogemuspõhise meetodi töövahendi esmase versiooni kirjeldusele [6] ja rakenduse kasutusjuhendile[1]. Kasutatavuse kogemuspõhise analüüsi (ingl *Experience-based Analysis*, EbA) meetod on kasutatavuse inspekteerimise meetod. EbA meetodit kasutatakse digitoote või e-teenuse nõuete analüüsi faasis, kohe, kui uus lahendus on kirjeldatud, et mõista, kas plaanitud lahendus hakkab toimima ja mis võivad olla lahendusega kaasnevad riskid kasutajate käitumise ja harjumuste seisukohast. EbA meetod vaatleb sidusrühmade varasemaid käitumisharjumusi ja aitab aru saada, kas sidusrühmadelt eeldatud teadmised, oskused, harjumused ja vahendid on piisavad planeeritava toote kasutamiseks. Analüüs aitab mõista juba enne prototüübi loomist, kas toote nõuded on kasutajate tegeliku käitumise ja harjumustega kooskõlas ning kas mingid aspektid vajavad põhjalikumat analüüsi. EbA meetod aitab leida nõuetes kohad, mida on vaja muuta või täiendada, muutes seeläbi kogu arenduse protsessi kiiremaks ja kuluefektiivsemaks. EbA meetodi komponendid ja analüüsi protsessi etapid on kirjeldatud joonisel 1.

[1] EbA-UIM rakendus. https://bitbucket.org/eba-method/eba-code/downloads/



Joonis 1. EbA meetodi raamistik[2]

EbA meetodi läbimiseks vajalikud elemendid ja etapid on lühidalt kirjeldatud järgnevas loetelus.

1. Jagage arendatav toode funktsioonideks (ingl *feature*) ehk väiksemateks osadeks. Iga funktsioon võiks sisaldada toote jaoks iseseisvat eesmärki või näidata, mis väärtust see toote jaoks loob.
2. Valige igale funktsioonile sidusrühm(ad) (ing *stakeholder*), mis on seotud konkreetse funktsiooniga. Samuti leidke eeltingimused, mis on vajalikud selle funktsiooni korrektseks toimimiseks.
3. Leidke igale eeltingimusele selle kontekstiga kattuv näide, mis oleks võimalikult sarnane situatsiooniga, mida valitud sidusrühm on varem kogenud ja mille põhjal saab analüüsida sidusrühma oskusi, harjumusi ja olemasolevaid vahendeid.
4. Hinnake, kui võrreldavad on sidusrühma perspektiivist eeltingimus ja näitena toodud sarnane situatsioon ja kas sidusrühm tegutseb näite situatsioonis ettenähtud ja oodatud viisil ning ega sidusrühmal ei ole kaebusi. Hindamisel vastake küsimustele: „Sama sidusrühm?“, „Sama kontekst?“. „Tegevus eesmärgikohane?“ ja „Sidusrühma rahulolu?“.
5. Kui olete ühele eeltingimusele toonud välja mitu näidet, siis soovi korral prioritiseerige ja valige välja kõige relevantsem näide.
6. EbA meetod koostab vastuse, mis oma kombinatsiooniga annab aimu, milline oleks olukord analüüsimiseks võetud eeltingimusega juhul, kui arendus toimuks hetkel

[2] EbA-UIM rakendus. https://bitbucket.org/eba-method/eba-code/downloads/

kavandatud viisil. Vastus tugineb infol, kui ootuspäraselt käitus sidusrühm antud situatsioonis vastavalt eeltingimuses seatud ootustele.

7. Peale saadud vastuse analüüsimist looge tegevuskava.

EbA meetodi jaoks on loodud Microsoft Exceli arvutustabelil põhinev töövahend [6, 7]. EbA meetodi mugavamaks kasutamiseks on loodud eraldiseisev rakendus, mille arendamist viiakse läbi mitmes etapis. Arenduse esimest ja teist etappi kirjeldatakse järgmises kahes alapeatükis.

1.2 EbA töövahendi esimese etapi kokkuvõte

EbA-UIM rakenduse arendamise esimese etapi viisid varasemalt läbi Iris Kreinin [2] ja Karl Olaf Kuldmaa [3]. Rakenduse esimese etapi tulemus on vabavaraline ja selle koodihoidla on nähtav keskkonnas Github [2]. Iris Kreinin disainis ning arendas oma bakalaureusetöö raames EbA-UIM rakenduse eessüsteemi ning tagas rakenduse vastavuse veebisisu juurdepääsetavussuuniste (edaspidi WCAG, ingl *Web Content Accessibility Guidelines*) 2.2 versioonile [2]. Karl Olaf Kuldmaa arendas oma bakalaureusetöö raames EbA-UIM rakenduse tagasüsteemi, andmebaasi, osa eessüsteemist ning paigaldamis- ja installeerimisloogika Windows operatsioonisüsteemiga arvutitele [3]. Esimeses etapis valminud rakenduses on võimalik luua uusi projekte ja viia läbi analüüs vastavalt EbA töövahendi kasutamise protsessile [2]. Lisaks on olemas eraldi lehed EbA meetodi kirjeldusele, EbA töövahendi kasutamise juhendile ja töövahendi tiimi kirjeldusele [2]. Rakendust saab kasutada nii eesti kui ka inglise keeles [3].

Järgnev lõik võtab lühidalt kokku esimeses etapis valminud EbA-UIM rakenduse loomiseks kasutatud programmeerimiskeeled, raamistikud ja tööriistad. Rakenduse tagasüsteem on arendatud kasutades Java[3] programmeerimiskeelt koos Spring MVC[4] raamistikuga, mis sobib hästi veebirakenduste loomiseks, tänu mugavale HTTP päringute haldusele [3]. Rakenduse kompileerimiseks, pakendamiseks ja sõltuvuste haldamiseks on kasutusel ehitusinstrument (ingl *build tool*) Maven[5] [3]. Andmete salvestamiseks ja haldamiseks on kasutatud SQLite[6] andmebaasi, mis on mõeldud andmete lokaalseks hoiustamiseks eraldiseisvates rakendustes [3]. Andmebaasi struktuuri ja andmete muutmiseks kasutatakse migreerimissüsteemi Flyway[7], mis võimaldab seda teha läbi automatiseeritud skriptide [3]. EbA-UIM rakenduse eessüsteem

[3] Java. https://www.java.com/en/
[4] Spring. https://spring.io/
[5] Maven. https://maven.apache.org/
[6] SQLite. https://www.sqlite.org/
[7] Flyway. https://www.red-gate.com/products/flyway/

on arendatud kasutades programmeerimiskeelt TypeScript[8] ja märgistuskeelt HTML koos Angular[9] raamistikuga [3]. Rakenduse kasutajaliidese mitmekeelseks muutmine on saavutatud mooduliga i18n[10]. EbA-UIM on pakendatud kasutades tööriista Launch4j[11], mis pakendab rakendused Windows operatsioonisüsteemiga arvutitele käitataval kujul [3]. Installeerija loomiseks kasutati tööriista Inno Setup[12], mis pakendab Java keskkonna ning Launch4j-ga pakendatud rakenduse installeerijaks [3]. Esimeses etapis valminud EbA-UIM rakendus on kasutatav ainult Windows operatsioonisüsteemiga arvutites [3].

1.3 EbA töövahendi teise etapi eesmärk

Siinse bakalaureusetöö eesmärk on edasi arendada „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendust ehk viia läbi rakenduse EbA-UIM teine etapp. Esimese etapi tulemusena valmis töötav rakendus ja tooteomanik soovis jätkata teise etapi arendustega, et rakendust veelgi kasutajasõbralikumaks muuta ning EbA meetodi rakendamist kasutajate jaoks lihtsamaks ja arusaadavamaks teha. Esimese etapi läbiviijad ja tootejuht koondasid teise etapi jaoks nimekirja täiendusvajadustest ja teise etappi planeeritud funktsionaalsustest. Sellele nimekirjale põhinedes viiakse läbi rakenduse arenduse teine etapp. Teise etapi saab jagada neljaks:

1. Esimese etapi poolt üle antud olemasolevate ja uute funktsionaalsuste analüüsimine ja implementeerimine, et muuta rakenduse kasutamine kasutajale veelgi arusaadavamaks ja kergemaks.
2. Rakenduse optimeerimine, sest suuremahuliste tabelite töötlemisel esineb jõudluspiiranguid, mis raskendavad rakenduse kasutamist või teevad töötamise suuremahulistes tabelivaadetes äärmiselt aeganõudvaks.
3. Rakendusele paindlikuma ja platvormist sõltumatu paigaldus- ja installatsiooniloogika loomine, et lihtsustada kasutuselevõtu protsessi ja võimaldada kasutamine erinevates operatsioonisüsteemides, nagu Windows, macOS ja Linux.
4. Teises etapis valminud arenduse tulemuste testimine läbi kasutatavuse ja manuaalse testimise.

Eelpool loetletud tööülesannete jaotus tiimiliikmete vahel on välja toodud järgmises alapeatükis.

[8] TypeScript. https://www.typescriptlang.org/
[9] Angular. https://angular.dev/
[10] i18n. https://www.npmjs.com/package/i18n
[11] Launch4j. https://launch4j.sourceforge.net/
[12] Inno Setup. https://jrsoftware.org/isinfo.php

1.4 Teise etapi arendusmeeskond ja tööjaotus

EbA-UIM rakenduse teine etapp sooritatakse tiimitööna. Järgnev peatükk tutvustab lähemalt kõiki tiimiliikmeid ning siinse töö autori ja teiste tiimiliikmete ülesandeid. Lisaks siinse bakalaureusetöö autorile kuuluvad tiimi Gerdo Germann, Rahel Pettai ja Timo Kaasik. Tooteomaniku positsiooni täidab EbA meetodi looja Anu Piirisild. Gerdo Germann [8] ja Rahel Pettai [9] täidavad tööülesandeid oma bakalaureusetöö raames.

Gerdo Germann viib läbi käsitestimisi jooksvalt arenduse käigus, iga sprindi lõpus [8]. Samuti viib ta arendusprotsessi lõpus läbi kasutajatestimised (edaspidi tarkvara testija) [8]. Lisaks testimisele vastutab Gerdo Germann ka rakenduse macOS ja Linux operatsioonisüsteemidele üleviimise eest [8]. Rahel Pettai ülesanneteks on läbi viia nõuete analüüs digiligipääsetavuse aspektist, et arendatav rakendus vastaks WCAG nõuetele (edaspidi ligipääsetavuse analüütik) ning uute funktsionaalsuste loomine ja olemasolevate täiendamine eessüsteemis (edaspidi eessüsteemi arendaja) [9]. Timo Kaasik viib läbi nõuete analüüsi (edaspidi nõuete analüütik) ja sarnaselt Rahel Pettaile arendab ka eessüsteemi (edaspidi eessüsteemi arendaja). Anu Piirisild on loonud arenduse teisele etapile nõuete kogumi, koostab rakenduses kuvatavad sisutekstid ja täidab tootearendusprotsessis tooteomaniku rolli (edaspidi tooteomanik).

1.4.1 Töö eesmärgid

Siinse töö autori ülesanneteks EbA-UIM rakenduse teises etapis on taga- ja eessüsteemi edasiarendamine, rakenduse platvormist sõltumatuks muutmine ning tiimitöö juhtimine. Siinne peatükk tutvustab nende ülesannete täpsemat sisu.

Tagasüsteemi edasiarendus hõlmab endas projektide alla- ja üleslaadimise loogika loomist. Autori ülesandeks on samuti eessüsteemi arendajatele tagasüsteemi toe pakkumine vastavalt nende vajadustele. Viimane hõlmab endas andmebaasi ja tagasüsteemi otspunktide kohandamist vastavalt uutele funktsionaalsustele.

Eessüsteemi edasiarendus hõlmab endas peamiselt eessüsteemi optimeerimist. Suuremahuliste tabelite puhul muutub rakendus aeglaseks ja seda tuleb optimeerida. Teise etapi arenduse alguses lepiti kokku, et lisaks rakenduse optimeerimisele teeb siinse töö autor vajadusel eessüsteemis väiksemaid programmivigade parandusi ja viib läbi väiksemamahuliste funktsionaalsuste rakendamist.

Rakenduse platvormist sõltumatuks muutmine tähendab esiteks erinevate olemasolevate raamistike uurimist ning teiseks koostöös Gerdo Germanniga uurimistulemuste võrdlemist,

siinse projekti jaoks parima raamistiku valimist ning olemasoleva projekti üleviimist valitud raamistikule. Siinse töö autor vaatleb võimalusi võtmes, et rakendus töötaks Windows operatsioonisüsteemidel. Gerdo Germann jälgib, et rakendus töötaks Linux ja macOS operatsioonisüsteemidel.

Tiimitöö juhtimine hõlmab endas iganädalaste koosolekute läbiviimist, kus antakse tooteomanikule ülevaade eelneva nädala arendusest ning luuakse plaan järgmiseks nädalaks. Samuti vastutab siinse töö autor ka arenduskeskkonna ülesseadmise eest ehk projektihaldustarkvara ja projektijuhtimismetoodika valimise ning nende rakendamise eest EbA-UIM rakenduse arendamise teises etapis.

1.5 Olemasoleva tarkvaraprojekti ülevõtmine

Käesoleva EbA-UIM rakenduse arenduse puhul on tegu olukorraga, kus teise etapi arendusmeeskond võtab üle eelmise etapi arenduse. Olemasoleva ehk kellegi teise alustatud tarkvaraprojekti ülevõtmine võib minna sujuvalt, kuid võib kujuneda hoopis keeruliseks ja ajamahukaks. Käesolev alapeatükk tutvustab uude tarkvaraprojekti sisenemise raskusi ning tegureid, mis võivad aidata uutel arendajatel kiiremini projekt omaks võtta.

Järgnev lõik võtab kokku leiud Barthélémy Dagenaisi jt kvalitatiivuurimusest [10], kus vaadeldi uue arenduskeskkonna või projektiga liitunud arendajaid. Selles uurimuses jälgiti 18 äsja liitunud arendajat 18 erinevas projektis ning prooviti leida, mis on kõige suurematekstakistusteks ja mis aitab kaasa uutel tulijatel uut keskkonda omaks võtta. Barthélémy Dagenaisi jt uurimistulemused näitasid, et uutel töötajatel võttis aega, et saada aru projekti arhitektuurist ja ärinõuetest, mõnel juhul põhjustasid raskusi ka uued tehnoloogiad, mida ei olnud arendajad varem näinud. Igal tiimil on tarkvaraarenduses oma lähenemine ja sellest arusaamine võttis samuti aega. Kõige raskemaks ülesandeks aga osutus arenduskeskkonna ülesseadmine – vajalike tööriistade allalaadimine, konfiguratsiooni tegemine ja projektihaldustarkvaraga tutvumine. Kõik uuritud projektid kasutasid agiilset arendusmetoodikat, mis aitas uutel arendajatel olla kohe kogu protsessi kaasatud, seega saadi osaleda planeerimises, arendamises ja ka testimises. Eriti aitas kaasa Scrum metoodika, mis näeb ette töö tegemiseks igapäevased koosolekud. Uuringus leiti kolm kõige tähtsamat tegurit, mis aitasid uutel arendajatel kiiremini projekti mõistma hakata. Esimeseks teguriks on kohe varakult koodiga eksperimenteerimine. Väiksemate ülesannete täitmine iseseisvalt katse-eksitus meetodiga andis uutele tulijatele kiiremini parema arusaama projektist, kui seda tegi dokumentatsiooni lugemine. Teiseks teguriks on ajapikku erinevate projekti struktuuride ja uue töökultuuri omaksvõtmine.

Teadmised koodi kirjutamise tavadest ja töökultuurist on tavaliselt töötajate seas vaiketeadmised. Selliste teadmiste omandamiseks peab uus töötaja tuginema tihedale suhtlusele kolleegidega või koodis leiduvatele vihjetele nagu detailne versiooniajalugu ja koodinäited. Kolmandaks teguriks on järelvalve ja sagedased edenemisaruanded kaastöötajatelt, et nii uuel töötajal kui ka tema kolleegidel oleks ülevaade, kas liigutakse lahendustega õiges suunas.

1.6 Platvormist sõltumatu rakenduse raamistik

Selle alapeatüki eesmärk on tutvustada ja võrrelda erinevaid olemasolevaid platvormist sõltumatuid (ingl *cross-platform*) töölauarakenduste (ingl *desktop application*) raamistikke ning kaaluda erinevaid võimalusi EbA-UIM rakenduse komplekteerimise, lihtsa allalaadimise ja paigalduse jaoks erinevatel operatsioonisüsteemidel.

Ishan Herath toob oma uurimistöös [11] välja, et tänapäeval on platvormist sõltumatute rakenduste arendamine tähtsam kui kunagi varem. Ishan Herathi sõnul oodatakse tänapäeval, et rakendused töötaksid sujuvalt kõikidel platvormidel, kõikide operatsioonisüsteemidega. Erinevad kogu koodi haldavad raamistikud lubavad kirjutada koodi vaid kord ja rakendada seda erinevatel platvormidel, nagu mobiil, veeb ja lauaarvuti, muutes kogu arendamise protsessi arendaja jaoks lihtsamaks ja kiiremaks [11].

EbA-UIM rakenduse esimeses etapis kasutusele võetud Launch4j[13] raamistik ei ole platvormist sõltumatu, see on loodud ainult Windows operatsioonisüsteemile. Lisaks avaneb rakendus veebilehitsejas nagu veebirakendused, mitte eraldiseisvas aknas nagu töölauarakendused. Järgnevalt võrreldakse erinevaid platvormist sõltumatuid töölauarakenduste raamistikke nagu Electron[14], JavaFX[15] ja Flutter[16].

JavaFX-i ja Electroni raamistikke on laialdaselt uuritud ja kasutatud erinevates rakendustes. Abeer Alkharsi ja Wasan Mahmoudi uurimistöö [12] annab hea ülevaate nende tehnoloogiate erinevustest, eelistest ja puudustest. JavaFX põhineb Java programmeerimiskeelel, mis on tugevalt objektorienteeritud [12]. Electron on aga väga sarnane erinevatele veebirakendustes kasutatavatele raamistikele, mis kasutavad JavaScript-i, HTML-i ja CSS-i [12]. Electroni on kasutanud ettevõtted nagu Microsoft, Facebook, Slack ja Docker [12].

[13] Launch4J. https://launch4j.sourceforge.net/
[14] Electron. https://www.electronjs.org/
[15] JavaFX. https://openjfx.io/
[16] Flutter. https://flutter.dev/

Eelmainitud uurimistöös [12] arendati välja kaks identset rakendust, kasutades JavaFX-i ja Electroni, ning hinnati nende jõudlust ja arenduse lihtsust. Tulemustest selgus, et Electroniga arendatud rakendus oli kiirem ja kasutas vähem mälu CRUD-operatsioonideks kui JavaFX-ga loodud rakendus. Lisaks toodi välja, et Electron, tänu oma sarnasusele veebirakendustes kasutatavate raamistikega, pakub suurepärast võimalust veebiarendajatel ka töölauarakendusi arendada. JavaFX aga võimaldab rakendada traditsioonilisi objektorienteeritud programmeerimise põhimõtteid. JavaFX-i plussideks tõid nad välja ka, et JavaFX Scene Builder tööriist võimaldab luua kiiremini ja väiksema vaevaga rakendusele kasutajaliidest ning kuna JavaFX on Electroniga võrreldes palju vanem raamistik, siis leidub sellele põhjalikum dokumentatsioon ja suurem kogukonna tugi.

Daniyar Alymkulov toob oma töös [13] välja ka mõned miinused Electroni kasutamise kohta. Esiteks peab Electron pakkima oma rakendusse sisse Chromium mootori, et käivitada veebipõhist koodi, JavaScripti, HTMLi, CSSi. Selle tõttu võtavad Electroni rakendused palju kettaruumi, näiteks väheste funktsionaalsustega väiksed rakendused vajavad juba 100 MB kettaruumi. Samuti leidis Daniyar Alymkulov, et Electron nõuab palju arvuti jõudlust, nii mälu kui ka protsessori jõudluse osas. Suurem jõudluse kasutamine on arusaadav, arvestades, et tegemist on platvormist sõltumatu rakendusega ja seda ei saagi võrrelda platvormipõhiste rakendustega, nagu ainult Windows operatsioonisüsteemile loodud .NET raamistik. Daniyar Alymkulov tõi samas välja ka positiivseid aspekte, näiteks Electronile leidub tänu oma veebirakendustele sarnaneva ülesehitusele rohkesti erinevaid mooduleid ja laiendusi.

Flutter on samuti üks populaarne platvormist sõltumatu raamistik. Flutter on Google'i loodud avatud lähtekoodiga tööriistakomplekt, mis võimaldab luua mobiili-, veebi- ja töölauarakendusi ühest koodibaasist [14]. Lukas Dagne uuris Flutterit oma töös [15] peamiselt operatsioonisüsteemide iOS-i ja Androidi arenduse jaoks, kuid kuna tööriist ühildub ka töölauarakendustega, on Flutteri kasutamine kaalumisel siinses töös, EbA-UIM rakenduse teise etapi arendamisel. Lukas Dagne järeldab, et Flutter on tugev valik rakenduste arendamiseks, eriti oma kiiruse ja kaasaegsete tehnoloogiate tõttu, kuid selle integreerimine juba olemasolevatesse ja teiste raamistike peale ehitatud projektidesse võib olla ajamahukas. Lukas Dagne arvab, et Flutter on veel liiga noor raamistik ja sellel leidub palju kitsaskohti, kuid kuna Flutteri kogukond ja arendustiim töötavad aktiivselt nende lahendamise nimel, teeb see Flutterist tulevikus potentsiaalselt väga hea valiku rakenduste arendamiseks.

Elias Müller võrdles oma töös [16] Flutterit ja Electroni, sarnaselt Abeer Alkharsi ja Wasan Mahmoudi tööle, luues mõlema raamistikuga samasuguse töölauarakenduse. Elias Müller leiab, et Flutteri kasutamine töölauarakenduste arendamisel on endiselt varajases staadiumis. Flutteri kasutamise eeliseks on intuitiivsus ja väiksem koodimaht võrreldes teiste raamistikega nagu Electron. Lõpuks leidis Elias Müller aga sarnaselt Lukas Dagnele, et Flutter on veel liiga noor raamistik ja vajab aega arenemiseks ning hetkel tasuks tugevalt kaaluda teisi võimalusi töölauarakenduste arendamiseks.

1.7 Tarkvaraprojektide juhtimine

Järgnev alapeatükk tutvustab peamisi kasutatavaid platvorme koodi hoiustamiseks ja projektihalduseks. Tutvustatakse ka erinevaid projektijuhtimise metoodikaid.

1.7.1 Projektihaldustarkvara

Versioonihaldus ja projektihaldustarkvarade kasutamine on kaasaegse tarkvaraarenduse lahutamatud osad. Need lihtsustavad oluliselt arendustiimide koostööd, võimaldades hallata koodimuudatusi ja ülesannete jaotust arendustiimi liikmete vahel. Käesolevas peatükis keskendutakse Git-versioonihaldussüsteemile ning sellele tuginevatele koodi hoiustamise platvormidele, nagu GitHub, GitLab ja Bitbucket.

Git-i mõistmiseks kirjutasid Scott Chacon ja Ben Straub raamatu „Pro Git“ [17]. Järgnev Git-i selgitus tugineb raamatule „Pro Git“ [17]. Git on vabavaraline versioonihaldustarkvara, mis on loodud aastal 2005 Linus Torvaldsi poolt. Esialgselt on Git loodud Linux operatsioonisüsteemi tuuma arendamiseks. Git hoiustab faile ja nende kogu muutuste ajalugu ühes repositooriumis (ingl *repository*) ehk ühe kausta sees, tagades võimaluse igal hetkel oma muudatused tagasi võtta ja jätkata eelmiselt versioonilt. Git on hädavajalik tööriist tiimidele, kes soovivad samaaegselt arendada sama koodibaasi.

Järgnev lõik tutvustab erinevaid koodi hoiustamise platvorme, tuginedes Salah Uddini jt artiklile [18]. Salah Uddin jt võrdlesid platvorme GitHub, GitLab, Bitbucket, AWS CodeCommit, Azure DevOps ja SourceForge. Enim kasutatuks platvormiks aastal 2022 osutus GitHub, millele järgnesid GitLab ja Bitbucket. Põhjalikumalt analüüsisid Salah Uddin jt esikolmikut, tuues välja iga platvormi pakutavad teenused, mis osutusid enamjaolt samaks kõikidel platvormidel. Võrdluses ei võetud arvesse Bitbucketi integreeritust Jiraga, mis lisab Bitbucketi baasteenustele veel Jira projektihaldusvõimalused.

Bitbucket on Atlassiani loodud Git-il põhinev koodi hoiustamise tööriist, mis on mõeldud kasutamiseks meeskondadele [19]. Bitbucket pakub erinevaid vahendeid koodiga töötamiseks, mille hulka kuuluvad võimalus koodi läbivaatuseks (ingl *code review*) ja CI/CD ehk pidevintegratsiooni ja pidevvalmiduse (ingl *continuous integration and continuous delivery*) lahendus, mille abil saab tarkvara testimise ja evitamise automatiseerida [19]. Lisaks on Bitbucketiga integreeritud tööriist Jira [19]. Jira on Atlassiani loodud agiilse projekti haldamise tööriist, mida kasutatakse tarkvaraarendusprotsessi planeerimiseks ja jälgimiseks, samuti ka tarkvara väljaandmise ja sellele toe pakkumise koordineerimiseks tiimides [20].

Üldjoontes pakuvad erinevad koodi hoiustamise platvormid ja projektihaldustarkvarad sarnaseid tööriistu.

1.7.2 Projektijuhtimise metoodika

Traditsioonilised arendusmetoodikad eeldavad, et kogu projekti plaan on enne arendusetappi täielikult defineeritud [21]. Kõige kasutatum traditsiooniline metoodika on loodud Dr. Winston W. Royce'i poolt ja selle nimeks on koskmudel (ingl *Waterfall model*) [21]. Koskmudeli etapid on järgmised: nõuete analüüs, disain, arendus, testimine ja hooldus [22]. Koskmudelis ei minda edasi järgmisse etappi enne kui eelmine etapp on täielikult läbitud ja juba läbitud etappi tagasi ei pöörduta [22]. Tihti on tiim jaotatud etappide järgi väiksemateks töögruppideks, näiteks analüütikud, disainerid, arendajad, testijad. Selline tiimijaotus põhjustab olukorra, kus erinevad inimesed töötavad suure projekti väiksemate osadega iseseisvalt ja eri aegadel ning töögrupid suhtlevad omavahel vähem kui agiilsete arendusmetoodikate korral [22]. Vastupidiselt agiilsetele metoodikatele antakse koskmudeli puhul toode kliendile üle projekti lõpus, seega ei saa arendustiim kliendi tagasisidet oma tööle enne, kui terve projekt on valmis [23].

Traditsiooniliste arendusmetoodikate kõrvale on tekkinud agiilsed lähenemised, mis rõhutavad paindlikkust, koostööd ja kiiret reageerimist muutustele. Nimi „agiilne" sündis aastal 2001, kui 17 tarkvaraarendajat kogunesid Utah's [24]. Nad esindasid kõik erinevaid arendusmetoodikaid, nagu ekstreemprogrammeerimine (ingl *Extreme Programming*), Scrum, DSDM jpt [24]. Kogunemise eesmärgiks oli leida nende alternatiivsete metoodikate ühine keel ning arutelu tulemusena valmis tarkvara välearendus manifest (ingl *Manifesto of Agile Software Development*) [24]. Gumiński jt kirjutasid oma artiklis [25], et üha enam kogub populaarsust Scrum, mis on üks agiilse tarkvaraarenduse metoodika vorme. Artiklis toodi välja, et paljud tarkvaraarendajad ja ka projektijuhid usuvad, et Scrum parandab meeskonnatööd, suhtlust ja aitab kaasa projekti õnnestumisele, kusjuures projekti õnnestumine tuleneb jooksvast

tarkvaratestimisest projekti käigus. Agiilsed metoodikad soosivad tihedat suulist suhtlemist, mis tõstab usaldust ja üksteise mõistmist kolleegide vahel [25]. Samuti annab agiilne metoodika arendustiimile suurema otsustusvõime ja valikuvabaduse kuidas ja mida teha ning vajadusel lubab ka varasemalt tarkvarale seatud nõudeid muuta [25]. Paljud tiimid nõustuvad, et võrreldes koskmudeliga, mis on üks klassikalistest arendusmetoodikatest, toob Scrum metoodika kaasa paremad suhted klientidega [25]. Seda kõike toetab veel ka Alzeyani ja Szabó uurimus agiilsete tarkvaraarendusmetoodikate efektiivsuse ja Scrum metoodika kohta projektijuhtimises [26]. Uurimuse tulemusena leiti samuti, et kui tarkvaraprojektis kasutada agiilset arendusmetoodikat, õnnestub projekt suurema tõenäosusega [26].

Iga projekt on erinev, kõigil on oma eeldused ja oma soovitav lõpptulemus. Erinevaid arendusmetoodikaid saab kombineerida või muuta vastavalt oma vajadustele ja vastavalt oma projektile – selliseid uusi metoodikad nimetatakse hübriidmudeliteks (ingl *hybrid model*) [23]. Tüüpiline hübriid traditsioonilise ja agiilse metoodika vahel näeb välja nii, et analüüs ja planeerimine sooritatakse projekti alguses nagu seda tehakse koskmudeli puhul, kuid arendusprotsess, testimine ja hooldus läbitakse iteratiivselt või tsükliliselt nagu seda tehakse agiilsete metoodikate puhul [23].

2. EbA töövahendi teise etapi teostus

Järgnev peatükk tutvustab EbA-UIM rakenduse teise etapi arendusprotessi: tagasüsteemi edasiarendust, baasi struktuuri uuendamist, installeerimis- ja allalaadimisloogika muutmist, tiimitöö sujuvust ja juhtimisprotsessi.

2.1 Rakenduse arendamine

Töö autor lõi tagasüsteemi võimaluse rakendusest projekte alla laadida Exceli formaadis, mis võimaldab kasutada EbA meetodi tulemusi näiteks aruannetes või saata tulemust edasi ka neile, kellel puudub EbA-UIM rakendus. Exceli faili genereerimiseks Java objektidest kasutas töö autor teeki Apache POI[17], mis võimaldab Microsoft Office'i dokumente lugeda, kirjutada ja muuta. Apache POI teegil on aktiivne tugi ja on laialt kasutatud, mis tähendab, et sellele leidub laialdaselt kasutusjuhendeid. Apache POI võimaldab luua uue Exceli faili, defineerida uusi lehti ja täita need andmetega. Lahtreid on võimalik disainida vastavalt vajadusele. Tooteomaniku edastatud nõuetele vastavalt sai loodud ka disain, milles on kasutatud samu värvikombinatsioone nagu EbA-UIM rakenduses. Näide genereeritud Exceli failist on nähtav lisas 1.

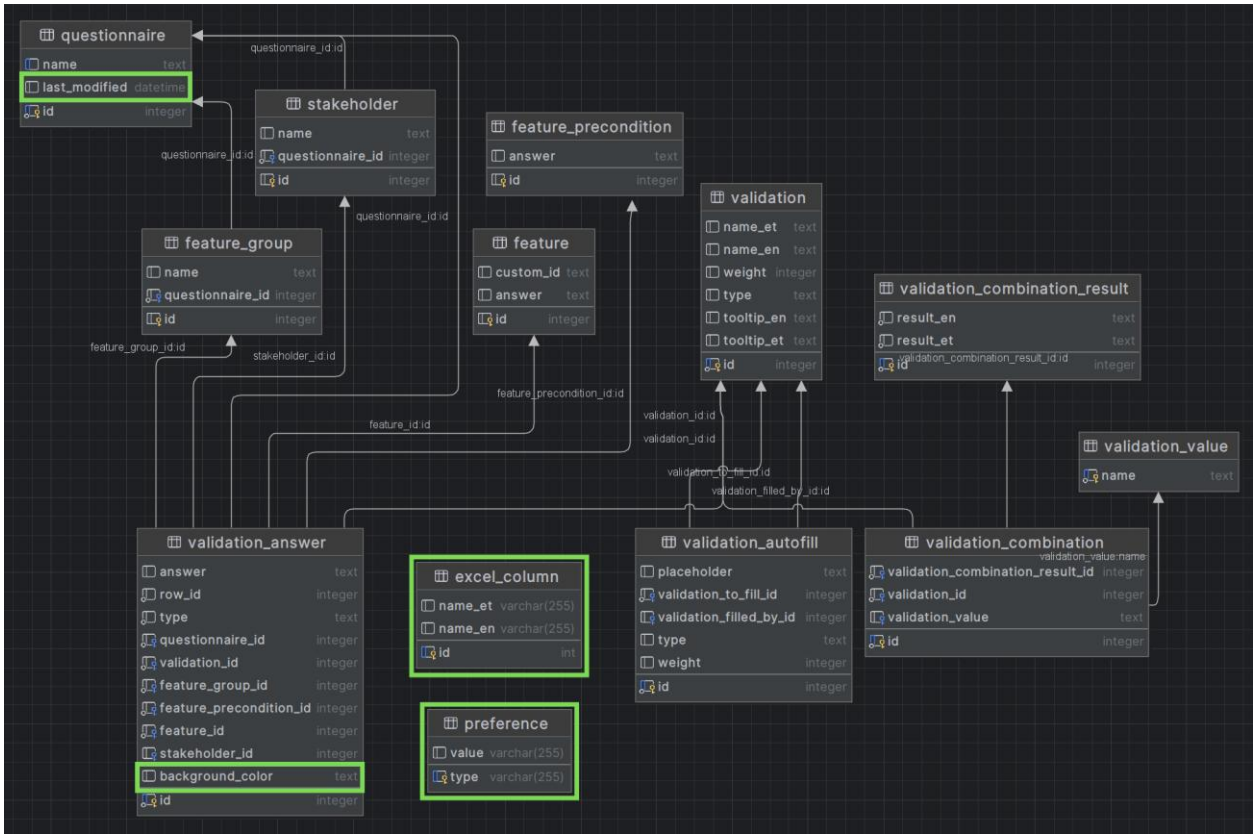
Vastavalt tooteomaniku esitatud nõuetele pidi saama projekte ka jagada erinevate seadmete vahel. Selleks tuli töö autoril luua võimalus projekti JSON-formaadis alla- ja üleslaadimiseks. Java objektide JSON-iks muundamisel kasutas autor Java teeki nimega Jackson[18], täpsemalt selle teegi klassi *ObjectMapper*. See on lihtne moodus muuta Java objekt JSON-kujul olevaks tekstiks. Samamoodi toimiti ka üleslaadimisel: JSON-fail muundati Java objektiks ja salvestati uued andmed andmebaasi. Uute andmete salvestamisel andmebaasi pidi jälgima, et andmetele ei jääks külge mitte vana, vaid uus automaatselt genereeritud identifikaator.

Siinse töö autor pakkus ka tagasüsteemi tuge eessüsteemi arendajatele, et tagada neile vajaminevad API otspunktid ja andmebaasi uuendused uute funktsionaalsuste lisamiseks. Nende hulka kuulusid näiteks andmebaasi uue tabeli loomine kasutaja eelistuste salvestamiseks, loogika projekti viimati muutmise kuupäeva ja kellaaja salvestamiseks ning kõigele eelpool nimetatule, kas olemasolevate otspunktide modifitseerimine või uute loomine.

Andmebaasi muutmisel kasutati migreerimissüsteemi Flyway ja selle automaatseid skripte. Töö autor lisas andmebaasi projektide tabelisse uue välja *last_modified* viimati muutmise

[17] Apache POI. https://poi.apache.org/
[18] Jackson ObjectMapper. https://jenkov.com/tutorials/java-json/jackson-objectmapper.html

kuupäeva salvestamiseks ja kasutajate vastuste tabelisse tulba *background_color* lahtrite taustavärvi salvestamiseks. Samuti lisas töö autor kaks uut tabelit. Esiteks sai lisatud tabel kasutajate eelistuste salvestamiseks, näiteks rakenduse keel. Teiseks sai lisatud tabel genereeritava Exceli tulpade eesti- ja ingliskeelsete nimede salvestamiseks. Andmebaasi diagramm on näha joonisel 2. Diagrammil on autori tehtud muudatused märgistatud rohelisega.



Joonis 2. Andmebaasi diagramm.

Autor lisas tagasüsteemi ka koodivormindaja nimega Spotless[19], mis on vabavaraline koodivormindaja. Spotless suudab vormindada mitmeid erinevaid programmeerimiskeeli, mille hulka kuulub ka Java. Autor nägi vajadust eraldiseisva vormindaja järele, et EbA-UIM rakenduse tagasüsteemi lähtekood oleks kooskõlas Java koodi heade tavadega. Olenevalt seadistusest aitab Spotless automaatselt eemaldada koodist kasutamata impordid, poolitada liiga pikad koodiread ja ühtlustada reataande pikkused üle kogu projekti. Spotless muutis koodi paremini loetavaks.

[19] Spotless. https://github.com/diffplug/spotless

2.2 Electron

EbA-UIM rakenduse platvormist sõltumatuks muutmise ning lihtsa allalaadimise ja paigalduse võimaldamiseks nii Windows, macOS kui ka Linux operatsioonisüsteemidel viidi rakendus üle uuele raamistikule. Peatükis 1.6 analüüsitud erinevate raamistike seast leiti olevat EbA-UIM rakendusele sobivaim Electroni raamistik. Valik Electroni kasuks tehti seetõttu, et esimeses etapis valminud rakendus on oma loomult väga sarnane veebirakendusega, kasutades Java tagasüsteemi ja Angulariga seotud JavaScript-il, HTML-il ja CSS-il põhinevat eessüsteemi. Electron pakub tuge just sellise rakenduse platvormist sõltumatuks muutmiseks, komplekteerimiseks ja erinevatel platvormidel installeerimiseks. JavaFX ja Flutter jäid kõrvale peamiselt seetõttu, et nad ei ühildu juba olemasoleva projektiga ja eeldaksid kogu rakenduse eessüsteemi ümber kirjutamist nõutavale kujule ja keelele. Samuti sai JavaFX-i välistamise argumendiks JavaFX-i vanus, sest on aru saada, et see on juba aegunud raamistik ja paljud arendajad on uuemate raamistike kasuks otsustanud. Vastupidisele JavaFX-ile on Flutter hiljuti loodud raamistik, kuid vajab veel aega, et konarused saaks eemaldatud. Flutteri raamistik võib aga tulevikus osutuda suurepäraseks valikuks platvormist sõltumatu tarkvara arendamiseks.

Siinse töö autor koostöös Gerdo Germanniga viis EbA-UIM rakenduse vanemalt Launch4J raamistikult üle uuemale ja platvormist sõltumatule Electroni raamistikule. Üldjoontes oli üleviimise protsess lihtne, kuna oli võimalik tugineda Electroni ametlikule juhendile[20] ja erinevatele interneti foorumitele. Electronile üleviimisel kujunes kõige ajakulukamaks etapiks Electroni käivitusfaili *main.js* korrektne seadistamine. Electroni käivitusfail on rakenduse sisenemispunkt ning määrab, kust ja kuidas hakkab Electron rakenduse komponente laadima ning käivitama. Käivitusfaili seadistamine vajas head ülevaadet kogu projekti arhitektuurist. Arvestama pidi näiteks sellega, et EbA-UIM rakenduse eessüsteem on arendatud TypeScripti keeles ning oluline oli suunata Electron juba JavaScriptiks kompileeritud failide juurde, mitte TypeScriptis oleva lähtekoodi juurde.

EbA-UIM rakendusel on Java tagasüsteem. Electron aga ei toeta Java rakenduste pakendamist ja käivitamist. Selleks, et Electron oskaks ka Java tagasüsteemi käivitada, pakkis töö autor selle JAR-failiks, mida sai mugavalt teha kasutades Maveni ehituse elutsüklit (ingl *Build Lifecycle*) pakenda (ingl *package*). See teeb võimalikuks Java-põhise tagasüsteemi käivitamise Electronile. Tagasüsteem käivitatakse eraldiseisva protsessina paraleeleselt Electroni graafilise liidesega ning protsess peatatakse peale graafilise liidese sulgemist.

[20] Electroni juhend. https://www.electronjs.org/docs/latest/tutorial/quick-start

Siinse töö autor oli vastutav selle eest, et EbA-UIM rakendus töötaks Windows operatsioonisüsteemidel. EbA-UIM rakenduse pakendamiseks ja Windows operatsioonisüsteemidele sobiva installeerija loomiseks kasutati Electron Forge'i. Leidus ka teisi võimalusi, näiteks Electron Builder, kuid valik tehti Electrron Forge'i kasuks, sest see oli soovituslik tööriist Electroni ametlikus juhendis. Installeerija loomisel ja pakendaja seadistamisel lähtuti peamiselt Electron Forge'i ametlikust juhendist[21]. Kuna Electron Forge pakendab kogu rakenduse üheks ASAR-failiks, siis tuli anda rakenduse JAR-kujul olev tagasüsteem sellele kaasa kui lisaressurss, mis jääb peale installeerimist algsele kujule. Tähtis on, et JAR-failide käivitamiseks on vaja süsteemi JRE-d (Java Runtime Environment) ja igal tavakasutajal seda ei pruugi olla. Seega tuli rakendusse kaasa pakkida JRE, mille abil käivitada JAR-faile, et tagada töövõime igas süsteemis. Sarnaselt tagasüsteemile pakiti JRE kaasa kui lisaressurss.

Esimeses etapis loodud rakendus andis installeerimisel kasutajale kaasa andmebaasi faili, mis sisaldas eesti- ja ingliskeelset näidisprojekti. Tulenevalt rakenduse kohandamisest Linux ja macOS operatsioonisüsteemidele tekkis vajadus andmebaasi faili salvestama hakata kasutaja dokumentide kaustas. Selle vajaduse põhjust saab lugeda täpsemalt Gerdo Germanni bakalaureusetööst [8]. Näidisprojektide kasutajale kaasa andmine lahendati kasutades uut projekti üleslaadimise funktsionaalsust. Kasutajal on võimalus alla laadida nii eesti- kui ka ingliskeelne näidisprojekt Bitbucketist JSON-failina ning üles laadida see oma installeeritud EbA-UIM rakendusse.

Töö autor lõi tooteomanikuga konsulteerides EbA-UIM rakendusele ka lihtsa ikooni, mida kasutatakse nii installeerija, töölaua otsetee kui ka rakenduse enda ikoonina. Ikooni loomiseks kasutas autor programmi nimega Inkscape[22], mis on vabavaraline vektorjooniste tegemise tarkvara. Ikooni on kujutatud joonisel 3.

[21] Electron Forge. https://www.electronforge.io/import-existing-project
[22] Inkscape. https://inkscape.org/

Joonis 3. EbA-UIM rakenduse ikoon

Joonisel 3 kujutatud ikooni visuaalis kasutatakse sama värvipaletti nagu on kasutatud rakenduse kasutajaliideses.

2.3 Rakenduse optimeerimine

EbA-UIM rakenduse esimese etapi arenduse järgselt ilmnes mahukamate tabelite kasutamisel kiiruse probleem, mistõttu sai rakenduse optimeerimine üheks oluliseks ülesandeks rakenduse teise etapi arenduses. Kui üks projekt koosnes rohkem kui 15 reast, muutus rakendus kasutamatult aeglaseks. Rakenduse muutis aeglaseks see, et rakenduses on projekti vaates igal tabeli real oma dünaamiliselt muutuvad nupud, valikud ja andmed, mistõttu suuremate ridade arvu puhul muutus lehe DOM (*Document Object Model*) liiga suureks ehk HTML-elemente sai lehele liiga palju. Mahukas DOM põhjustab suurenenud mälu kasutust ning stiilitöötlus ja toimingute reageerimisvõime aeglustub. Selle probleemi lahendamiseks kasutas autor komponenti <*cdk-virtual-scroll-viewport*> Angulari CDK-st (*Component Dev Kit*)[23]. <*cdk-virtual-scroll-viewport*> on Angulari komponent, mis muudab pikkade nimekirjade või tabelite kuvamise veebilehel tõhusamaks. See laeb korraga ekraanile ainult elemendid, mis on parasjagu kasutajale nähtavad. EbA-UIM rakenduse kontekstis tähendab see seda, et mällu on laetud ja visualiseeritud korraga ainult need read, mis mahuvad kasutaja ekraanile. Selline virtuaalne visualiseerimine eeldab, et kõik nimekirja elemendid oleksid ühekõrgused. Kuna EbA-UIM rakenduse tabeli read olid siiani dünaamiliselt muutuva kõrgusega, tuli selle vormistust natuke muuta, et tabeli read oleksid alati ühekõrgused. Lisaks tekkis probleem HTML-atribuudiga *rowspan* ehk mitut rida üks tabelilahter läbib. Probleem tulenes sellest, et kui mitut rida läbiva lahtri algus ei olnud enam lehele laetud, „unustasid“ sellele järgnevad read, et see lahter eksisteeris ja kogu tabel nihkus selle lahtri võrra vasakule. Manuaalselt

[23] Angular CDK. https://material.angular.dev/cdk

tühjade lahtrite lisamine kohtadesse, mida eelnevalt täitis üks venitatud lahter, lahendas probleemi.

2.4 Tiimitöö ja rakenduse teise etapi arendustööde juhtimine

Järgnev peatükk tutvustab lähemalt, millised otsused tiimitöö ja projekti juhtimises tehti. Samuti antakse ülevaade, milline oli kasutatud projektihaldustarkvara ning projektijuhtimis-metoodika.

2.4.1 Projektihaldustarkvara

Autor valis EbA-UIM rakenduse arendustööde haldamiseks Bitbucketi ja Jira. Valik tulenes peamiselt autori varasemal positiivsel kogemusel tiimiprojektides, kus on kasutanud Bitbucketit ja Jirat.

Tooteomanik andis arendustiimile esialgsed nõuded tabeli kujul Google Docs failis. Peale nõuete analüüsi tõsteti analüüsitud nõuded edasi piletite kujul Jira keskkonda, kus oli neid arendajatel mugavam vaadata ja vajadusel väiksemateks osadeks jaotada. Iga funktsionaalsus, parandus ja jätkuarendus kategoriseeriti Jiras omakorda suurematesse ülesannete plokkidesse ehk eepostesse (ingl *epic*), et paremini eristada ülesannete olemust. Iganädalaste koosolekute ajal planeeriti tulevaks nädalaks sprint, mis hõlmas endas hetkel ajaliselt kõige kriitilisemate piletite valimist, nende sprinti tõstmist, igale piletile tegija määramist ja panuspunktide (ingl *story point*) ennustuse määramist, et ennustada sprindi raskusastet. Seda kõike sai väga mugavalt teostada Jiras. Arenduse käigus sai märkida ka Jira tahvlile piletite hetkeseisu, et kõigil tiimiliikmetel oleks ülevaade, millega igaüks hetkel tegeleb ja kui kaugel keegi oma tööülesannetega on. Selline tööülesannete seisu märkimine oli tiimis tähtis, sest nii sai testija teada, millal ja mida on vaja testida. Seisu märkimine käis Jiras pileteid tulpade vahel tõstes, seisude tulbad on kohandatavad vastavalt vajadusele. Siinse töö käigus olid kasutatud tulbad järgmised:

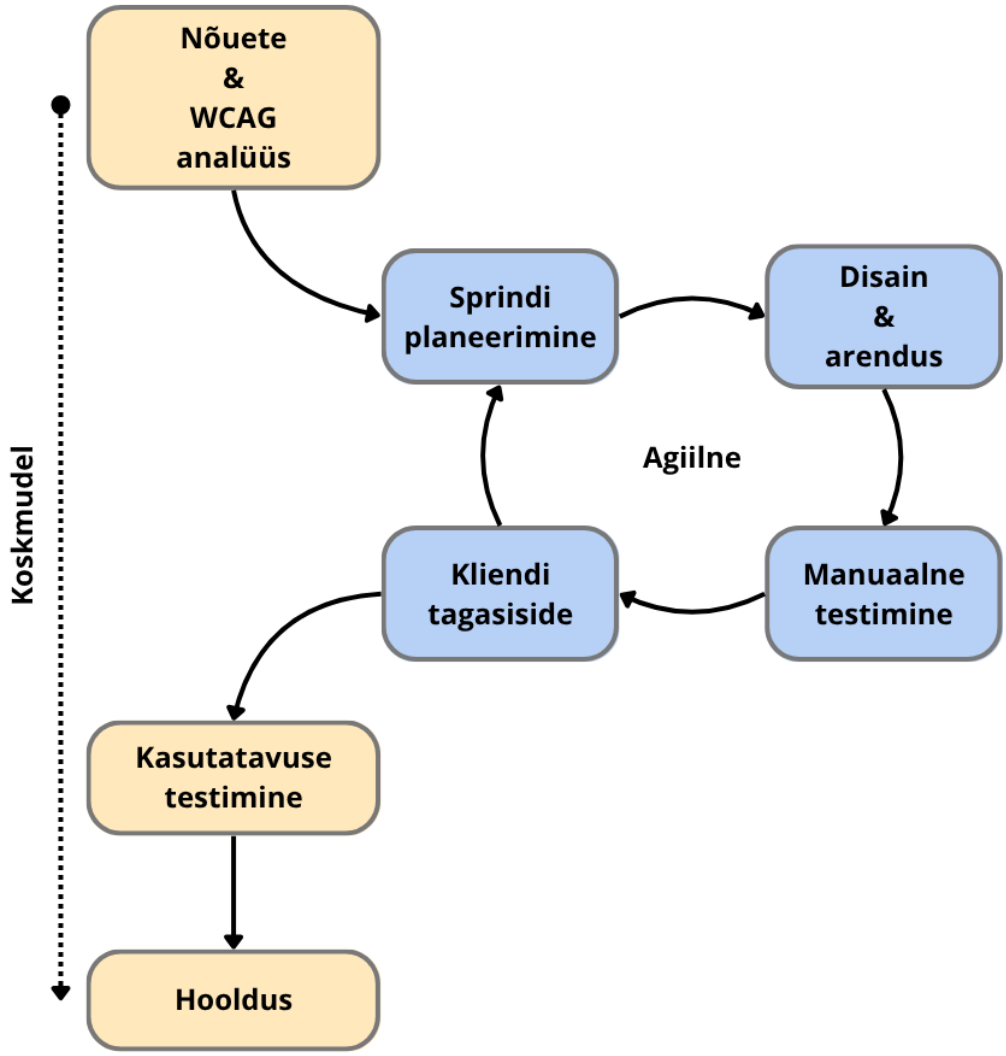
1. TO DO – keegi pole veel piletiga tegelema hakanud
2. DEVELOPMENT – pilet on arenduses
3. PULL REQUEST – pilet on arendatud ja kood ootab ülevaatamist
4. TESTING – tehtud muudatused on jõudnud arenduse harusse ja ootavad testimist
5. TESTED – tehtud muudatused on testitud
6. DONE – tehtud muudatused on jõudnud põhiharusse ja tooteomanikule on tulemust tutvustatud

Koodi hoiustamiseks oli kasutusel Bitbucket. Kuna Bitbucket ja Jira on mõlemad Atlassiani loodud, siis on nad omavahel integreeritud. Täpsemalt seotakse omavahel iga koodi kehtestus (ingl *commit*) ja Jira pilet, mis võimaldab hiljem mugavat versioonihaldust. Selliseks sidumiseks tuleb panna iga koodi kehtestuse päisesse Jiras automaatselt genereeritud pileti number ning piletile tekib automaatselt külge märge ja link, mis viib otse Bitbucketisse.

EbA-UIM rakenduse teise etapi töid alustades otsustati tiimis, et tõmbekutseid (ingl *pull request*) ja koodi ülevaatust tegema ei hakata, sest arendustiim on väike ja see tooks kaasa ebavajaliku ajakulu. Arenduse keskpaigas otsustas siinse töö autor ikkagi hakata tegema tõmbekutseid, et koodi kvaliteeti hoida. Koodi ülevaatus läbi tõmbekutsete tagas, et koodi ei jääks sisse ebavajalikke konsooli logimisi ja kommentaare ning et koodi efektiivsus ei langeks. Tõmbekutseid sai väga mugavalt teha Bitbucketis. Bitbucketis sai kommenteerida teise tiimiliikme koodi kehtestust ning nõuda vajadusel muudatusi. Kui muudatusi polnud vaja või need said parandatud, sai vajutada mugavalt nuppu „*merge*“ ehk mesti, mis mestis automaatselt märgistatud harud.

2.4.2 Projektijuhtimise metoodika

EBA-UIM rakenduse teise etapi arendusel kasutatakse hübriidmudelit, kus analüüs ja viimane testimine on kui koskmudeli etapid, kuid disain, arendus ja esialgne testimine teostatakse nädalastes sprintides nagu agiilsele metoodikale kohane. Otsus hübriidmudelit kasutada tulenes sellest, et jätkuarenduse nõuded ja skoop on juba arendusetapi alguses teada, seega saab nõuete analüüsi teostada juba enne arendustööde algust. Teisalt on arenduseks ettenähtud ajaraam lühike ning võib osutuda vajalikuks disainivalikute muutmine või nõuete kohandamine – seda võimaldab paindlik agiilne arendusmetoodika. Teise etapi arenduses tuleb nõuetele teha analüüs ja digiligipääsetavuse kriteeriumitele vastavuse analüüs, mille teostavad nõuete analüütik ja ligipääsetavuse analüütik enne arendustööde algust. Arenduse ajal toimuvad iganädalased koosolekud, millest võtavad osa kõik tiimiliikmed, sh ka tooteomanik. Koosolekutel arutatakse eelmise sprindi tulemusi ja antakse tooteomanikule ülevaade tehtust. Vajadusel tehti tooteomanikule ka demonstratsioon uutest lahendustest. Sellised koosolekud tagavad tooteomanikule võimaluse anda arendusprotsessi käigus jooksvalt tagasisidet arendatavale rakendusele. Kasutatavuse testimine oli juba projekti alustades plaanitud teostada kõige viimase etapina, peale arendustööde lõppu. Loodud hübriidse metoodika faasid EbA-UIM rakenduse teise etapi arenduseks on välja toodud ka joonisel 4.



Joonis 4. Hübriid koskmudeli ja agiilse arendusmetoodika vahel

2.4.3 Tiimijuhi positsioon

Siinse töö autori üheks ülesandeks oli ka tiimitöö korraldamine ja tehnilise toe pakkumine tiimikaaslastele. See hõlmas endas iganädalaste koosolekute korraldamist ja tiimiliikmete tehnilistele küsimustele vastamist, et hoida töö sujuvana. Regulaarsed iganädalased koosolekud toimusid Zoomi keskkonnas. Koosolekutel tegi autor kogu tiimile ja tooteomanikule kokkuvõtte viimasel nädalal tehtust ning väljakutsetest, kus andis sõna ka teistele tiimiliikmetele, enamasti arendajatele, kes said täpsemalt seletada, kuidas planeeritud ülesanded sujusid. Nädala kokkuvõtte ajal tõi testija välja ka manuaaltestimise tulemused.

Samuti viis töö autor tiimijuhina koosolekutel läbi järgmise nädala sprindi planeerimise ja võttis sprinti sisse uued piletid. Kui kõik eelmise nädala sprindis olnud ülesanded ei olnud lõpetatud, siis uuris töö autor põhjuseid ning tõstis pooleliolevad ülesanded järgmisesse sprinti. Töö autor andis tiimijuhina uute piletite sprinti võtmisel sõna ka nõuete analüütikule, kes seletas kogu tiimile, kuidas täpsemalt on plaanis nõue täita. Koosolekute lõppu jäi ka aeg, kus kõik tiimiliikmed said jagada oma täiendavaid tähelepanekuid, muresid ja positiivseid kogemusi.

Töö autor korraldas ka erakorralisi koosolekuid, seda enamasti ainult arendajate vahel. Need toimusid juhul, kui oli tekkinud projekti kohta tähtsamaid arhitektuurilisi või tehnilisi küsimusi, mida oli parem arutada omavahel. Autor viis läbi ka üks-ühele vestluseid tiimikaaslastega, kes mingil põhjusel ei saanud osaleda iganädalasel suurel koosolekul, et viia nad kurssi tehtud otsuste ja uue nädala sprindiga. Koosolekute välised vestlused tiimiliikmete vahel toimusid peamiselt Discordis, kus siinse töö autor vastas nii väiksematele kui ka suurematele tehnilistele küsimustele ja toetas tiimiliikmeid tööülesannete täitmisel.

3. Tulemus ja arutelu

Käesolevas peatükis tuuakse välja töö tulemused. Peatükis arutletakse ka peamiste tekkinud väljakutsete üle ning tuuakse välja valitud projektijuhtimise metoodika ja projektihaldustarkvara sobivus EbA-UIM rakenduse edasiarenduse teisele etapile.

3.1 Tulemused kasutajatega testimiselt

EbA-UIM rakenduse teise etapi arenduste käigus tehtud töödele sai töö autor tagasisidet tarkvara testija poolt läbi viidud kasutatavuse testimiselt, mille protsessi on kajastatud täpsemalt tema bakalaureusetöös [8]. Kasutatavuse testimist tehes jälgitakse, kuidas inimesed loodud tarkvara tegelikus olukorras kasutavad ning kuidas nad loodud funktsionaalsustele reageerivad [27]. Sellise testimise eesmärk on leida võimalusi tarkvara kasutajakogemuse parandamiseks. EbA-UIM rakenduse kasutatavuse testimine viidi läbi viie inimesega, kes kasutasid testimiseks oma isiklikku arvutit. Testimises kasutatud arvutite seas leidus nii Windows kui ka macOS operatsioonisüsteemiga arvuteid. Kasutajad pandi analüüsima mõnda oma projekti EbA-UIM rakenduses, kusjuures neile anti ka kindlaid ülesandeid, mida nad pidid rakenduses lahendama.

Testimise tulemused näitasid, et kõik tähtsamad arendatud funktsioonid töötasid nii nagu planeeritud. Peamised probleemid ilmnesid aga rakenduse installeerimisel ja disainilahendustes. Järgnevalt tuuakse välja testimise tulemused, mis puudutavad siinse töö autori tehtud arendusi. Rakenduse installeerimisel kuvavad nii Windows kui ka macOS operatsioonisüsteemid kasutajale turvahoiatust, mis tekitab turvahoiatusega varem mitte kokkupuutunud kasutajas kõhklust. Turvahoiatust kuvatakse, sest rakendusel pole veel kehtivat litsentsi. Windowsi operatsioonisüsteemis saab turvahoiatusest kasutajaliidest kasutades mööda, seega said testimisel kasutajad rakenduse allalaadimisega hakkama. MacOS operatsioonisüsteemis peab aga kasutama turvahoiatusest möödumiseks süsteemi konsooli. Rakenduse teise etapi raames litsentsi ei looda ja see jääb tooteomaniku hilisemaks ülesandeks. Peale testimist vastasid kasutajad küsimustikule. Väitele „Leian, et rakendust on lihtne alla laadida“ vastati viie punkti skaalal keskmiselt 4.4, kus vastus „1“ tähendas, et ei nõustu väitega ja „5“, et nõustun väitega.

Siinse töö autor lõi projekti alla- ja üleslaadimise loogika ning valis ka selle teostamiseks vajaminevate nuppude asukohad rakenduse kasutajaliideses. Kasutajatega testimiselt ilmnes, et projekti allalaadimise nupu leidmine tekitas vähesel määral segadust. Projekti allalaadimise kohta hakati otsima projekti vahekaardilt, mitte esilehelt projekti nime kõrvalt, kuid kui seda

projekti vahekaardilt ei leitud, mindi rakenduse avalehele ja sealt leiti see kerge vaevaga üles. Kasutajatele parema kogemuse loomiseks otsustati lisada allalaadimise võimalus ka projekti vaatesse, kuid selle arendus toimub järgmises arendusetapis. Projekti allalaadimine nii Exceli kui ka JSON-formaadis õnnestus testimise käigus igal kasutajal. Samuti õnnestus ka JSON-faili üleslaadimine.

Rakenduse töökiirus oli testimise ajal nõuetele vastav, kuid ilmnes projektitabeli vormistuse vigu. Näiteks projektitabeli viimase rea alumist piirjoont ei kuvatud ja horisontaalseks liikumiseks kasutatav kerimisriba asus väiksemate tabelite puhul liiga all või isegi ekraanist väljaspool. Need vead tulenesid otseselt rakenduse optimeerimisest ja sellest tulenevate tabeli vormingu muudatustest, mida on kirjeldatud peatükis 2.3. Töö autor muutis teise etapi arenduse käigus ka kerimisribade disaini ning testimiselt tuli välja, et kasutajad ei märka neid.

Kasutatavuse testimise käigus ilmnenud vigu ja parendamise võimalusi analüüsitakse ning otsustatakse, millised on prioriteetsemad ja lahendatakse veel teise etapi jooksul. Vähemprioriteetsed parandused suunatakse arenduse järgmisesse etappi.

3.2 Väljakutsed ja arutelu

EbA-UIM rakenduse teise etapi arendusprotsess esitas töö autorile mitmeid väljakutseid. Suurimaks väljakutseks osutus esimeses etapis loodud rakenduse üleviimine Electroni raamistikule. See samm oli eriti pingutustnõudev seetõttu, et siinse töö autor tutvus just selle sammu juures esimest korda põhjalikumalt esimeses etapis valminud lähtekoodiga ning pidi omandama väga hea ülevaate kogu rakenduse arhitektuurist. Nagu välja toodud peatükis 1.5 on just see – projektist ülevaate omandamine ning töökeskkonna ülesseadmine – üks väljakutsuvamatest sammudest uude tarkvaraprojekti sisenedes. Olukorra muutis keerukamaks ka see, et arenduse teises etapis ei osalenud ühtegi esimese etapi arendajat. Samuti oli see samm kogu arendusprotsessis väga kriitilise tähtsusega, sest enne kui rakendust ei ole võimalik testimiseks käivitada, on väga raske alustada uute funktsionaalsuste arendusega. Peale paljude juhendite ja foorumite lugemist, koostöös Gerdo Germanniga, oli EbA-UIM rakenduse üleviimine Electroni raamistikule edukas. Kuna tehnilise toe tagamine ja teiste tiimiliikmete abistamine nende arenduskeskkonna ülesseadmisel oli siinse töö autori üks tööülesannetest, edastas töö autor teistele tiimiliikmetele selged juhised, kuidas arenduskeskkond üles seada ja rakendust käivitada ning vastas jooksvalt täiendavatele küsimustele.

Töö autori poolt valitud projektijuhtimise metoodika ja projektihaldussüsteemid sobisid siinse EbA-UIM rakenduse edasiarenduse teisele etapile. Hübriid koskmudeli ja agiilse

arendusmetoodika vahel tagasid põhjaliku nõuete analüüsi juba enne arenduse algust, mille teostasid nõuete analüütik ja ligipääsetavuse analüütik koostöös tooteomanikuga. Tänu arenduse jaotamisele nädalastesse sprintidesse sai arendustiim tooteomanikult jooksvat tagasisidet tehtud töö kohta. Jooksvalt läbi viidud manuaalsed testimised, mida teostas tarkvara testija, aitasid leida arenduses varakult vigu ning tagas nende kiire lahendamise. Valitud projektihaldussüsteemides, milleks olid Bitbucket ja Jira, olid olemas kõik vajalikud tööriistad edukaks projekti lõpptulemuseks. Tänu Jira piletisüsteemile omasid kõik tiimiliikmed alati head ülevaadet, milliste ülesannetega tiimikaaslased tegelevad. Bitbucketi mugav koodi ülevaatuse võimalus aitas parandada koodi kvaliteeti. Bitbucket Downloads ehk lähtekoodist eraldatud sektsioon suuremate failide üleslaadimiseks võimaldas jagada tooteomanikuga EbA-UIM rakenduse uusimat versiooni.

Teises etapis valminud EbA-UIM rakendus on vabavaraline ehk lähtekood on avalik ja nähtav koodi hoiustamise platvormil Bitbucket[24]. EbA-UIM rakenduse installeerija nii Windows, macOS kui ka Linux operatsioonisüsteemidele on samuti kättesaadav platvormilt Bitbucket[25].

[24] EbA-UIM rakenduse lähtekood. https://bitbucket.org/eba-method/eba-code/src/master/
[25] EbA-UIM rakendus. https://bitbucket.org/eba-method/eba-code/downloads/

Kokkuvõte

Siinse bakalaureusetöö eesmärgiks oli viia läbi EbA-UIM rakenduse arendamise teine etapp. Töö autori ülesanneteks teises etapis olid taga- ja eessüsteemi edasiarendamine, EbA-UIM rakenduse platvormist sõltumatuks muutmine ja tiimitöö juhtimine.

Tagasüsteemi arendas töö autor juurde projektide Exceli formaadis allalaadimise ja JSON-failina alla- ja üleslaadimise loogika. Loogika loodi kasutades teeke Apache POI ja Jackson. Lisaks alla- ja üleslaadimise loogikale pakkus töö autor eessüsteemi arendajatele tagasüsteemi tuge, modifitseerides vajalikke API otspunkte ja andmebaasi tabeleid. Eessüsteemis optimeeris töö autor projektitabeli kuvamise, kasutades Angular CDK komponenti *<cdk-virtual-scroll-viewport>*, mis laeb korraga ekraanile ainult need nimekirja või tabeli elemendid, mida kasutaja reaalselt oma ekraanil näeb.

Vaadeldes erinevaid olemasolevaid raamistikke valis töö autor EbA-UIM rakenduse platvormist sõltumatuks muutmiseks raamistiku Electron. Samuti teostas töö autor koostöös Gerdo Germanniga arendatava rakenduse üleviimise eelnevalt kasutatud Launch4J raamistikult valitud Electron raamistikule. Kuna Electron ei toeta automaatset Java tagasüsteemi pakendamist ja käivitamist, pakkis töö autor tagasüsteemi JAR-failiks, kasutades Maveni ehituse elutsüklit. Töö autor lõi loogika tagasüsteemi käivitamiseks eraldiseisva protsessina paralleelselt Electroni graafilise liidesega. Windows operatsioonisüsteemile vajaliku installeerija loomiseks kasutas töö autor tööriista Electron Forge. Töö autor disainis ja lõi EbA-UIM rakendusele ka ikooni. Valminud vabavaralise EbA-UIM rakenduse lähtekood ja installeerijad on saadaval keskkonnas Bitbucket.

Tiimitöö juhtimine hõlmas endas projektihaldustarkvara ja projektijuhtimise metoodika valimist, iganädalaste koosolekute läbiviimist ning tehnilise toe pakkumist. Töö autor valis projekti haldamiseks Atlassiani loodud tarkvarad Bitbucketi ja Jira. Bitbucketit kasutati koodi hoiustamiseks ja tõmbekutsete läbiviimisseks. Jiras teostati sprintide planeerimine ja tööülesannete haldus. Töö autor valis arendamise metoodikaks hübriidi koskmudeli ja klassikalise agiilse arendusmetoodika vahel, mille faasid on nähtavad joonisel 4. Töö autor pakkus tiimikaaslastele tehnilist tuge, leides lahendusi erinevatele tekkinud tehnilistele küsimustele.

Töö autor sai bakalaureusetöö käigus rakendada oma teoreetilisi ja praktilisi oskusi reaalses tarkvaraprojektis. Töö käigus süvenes autor nii taga- kui ka eessüsteemi arendusse, õppides tundma uusi teeke ja raamistikke. Samuti arenesid autori oskused arendustöö juhtimises ja

meeskonnatöös. Kokkuvõttes sai töö autor arendava ja väärtusliku kogemuse läbi erinevate tarkvaraarenduse etappide teostamise.

Viidatud kirjandus

[1] Piirisild A., Perandrés Gómez A., Taveter K. A New Usability Inspection Method: Experience-Based Analysis. *Requirements Engineering: Foundation for Software Quality*. Cham: Springer Nature Switzerland, 2024, 74–91. https://doi.org/10.1007/978-3-031-57327-9_5 (08.12.2024).

[2] Kreinin I. Kasutajaliidese disaini loomine „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele. TÜ arvutiteaduste instituudi bakalaureusetöö. 2024. https://hdl.handle.net/10062/105053 (06.04.2024).

[3] Kuldma K. O. Rakendusprogrammi loomine „Kasutatavuse kogemuspõhise analüüsi meetodile“. TÜ arvutiteaduste instituudi bakalaureusetöö. 2024. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=79625 (10.04.2024).

[4] Cybernetica. Andmekaitse ja infoturbe leksikon. https://akit.cyber.ee/.

[5] e-Teatmik: IT ja sidetehnika seletav sõnaraamat. http://ww.vallaste.ee.

[6] Piirisild A. Tool for the Experience-based Analysis (EbA) method. 2024. https://doi.org/10.23673/re-453 (06.04.2024).

[7] Piirisild A. Analoogiatel põhinev tehnoloogia tulemuslikkuse ennustusmudel ja töövahend. TÜ ühiskonnateaduste instituudi magistritöö. 2021. http://hdl.handle.net/10062/72631 (06.04.2024).

[8] Germann G. Testimised ja macOSile kohandamine „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele. TÜ arvutiteaduste instituudi bakalaureusetöö. 2025.

[9] Pettai R. Kasutajaliidese disaini ja digiligipääsetavuse edasiarendus „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele. TÜ arvutiteaduste instituudi bakalaureusetöö. 2025.

[10] Dagenais B., Ossher H., Bellamy R. K. E., Robillard M. P., Vries J. P. Moving into a New Software Project Landscape. *In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. New York:

Association for Computing Machinery, 2010, p. 275-284. https://doi.org/10.1145/1806799.1806842 (10.04.2025)

[11] Herath I. Cross-Platform Development With Full-Stack Frameworks: Bridging the Gap for Seamless Integration. Jyväskylä University of Applied Sciences master's thesis. 2024. https://www.theseus.fi/handle/10024/867532 (08.12.2024).

[12] Alkhars A., Mahmoud W. Cross-Platform Desktop Development (JavaFX vs. Electron). Linnaeus University Faculty of Technology bachelor's thesis. 2017. diva2:1081105 (08.12.2024)

[13] Alymkulov D. Desktop Application Development Using Electron Framework Native vs. Cross-Platform. South-Eastern Finland University of Applied Sciences bachelor's thesis. 2019. https://www.theseus.fi/handle/10024/167669 (08.12.2024)

[14] Google. Flutter. 2024. https://docs.flutter.dev (08.12.2024).

[15] Dagne L. Flutter for cross-platform App and SDK development. Metropolia University of Applied Sciences Information Technology bachelor's thesis. 2019. https://www.theseus.fi/handle/10024/172866 (08.12.2024).

[16] Müller E. Web Technologies on the Desktop: An Early Look at Flutter. University of Stuttgart Institute of Architecture of Application Systems bachelor's thesis. 2021. http://dx.doi.org/10.18419/opus-11498 (08.12.2024).

[17] Chacon S., Straub B. Pro Git. 2nd Edition. Apress, 2014, Version 2.1.443 (09.03.2025).

[18] Salah Uddin M., Jahid Hasan K., Tasnima R. A Comparative Analysis of the Rise of Git-Based Distributed Collaborative Hosting Platforms: Survey, Performance Test, and Comparison". *Asian Journal of Engineering and Applied Technology*. The Research Publication, 2023, Vol. 12, No. 2, p. 29-33. https://doi.org/10.51983/ajeat-2023.12.2.3969 (09.03.2025)

[19] Atlassian Corporation. Bitbucket. 2025. https://bitbucket.org/ (09.03.2025).

[20] Atlassian Corporation. Jira. 2025. https://www.atlassian.com/software/jira (09.03.2025).

[21] Gaborov M, Karuović D, Kavalić M, Radosav D, Milosavljev D, Stanisaljev S, Bushati J. Comparative analysis of agile and traditional methodologies in IT project management. *Journal of Applied Technical and Educational Sciences*. 2021, Vol. 11, No. 4, p. 1-24. https://doi.org/10.24368/jates.v11i4.279 (09.03.2025)

[22] McCormick M. Watefall vs. Agile Metodology. MPCS Inc., Newburgh. 2012. http://mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf (09.03.2025)

[23] Fagarasan C., Popa O., Pisla A., Cristea, C. Agile, waterfall and iterative approach in information technology projects. *IOP Conference Series: Materials Science and Engineering*. Bristol: IOP Publishing, 2021, Vol. 1169, No. 1, p. 012025. https://doi.org/10.1088/1757-899X/1169/1/012025 (09.03.2025)

[24] A Manifesto for Agile Software Development. 2001. https://agilemanifesto.org/ (06.05.2025).

[25] Gumiński, A., Dohn, K., Oloyede, E. Advantages and Disadvantages of Traditional and Agile Methods in Software Development Projects – Case Study. *Organization and Management Series*. Katowice: Silesian University of Technology Publishing House, 2023, No. 188, p. 191–206. http://dx.doi.org/10.29119/1641-3466.2023.188.11. (09.03.2025).

[26] Alzeyani, E.M.M., Szabó, C. A Study on the Effectiveness of Agile Methodology Using a Dataset. *Acta Electrotechnica & Informatica*. 2023, Vol. 23, No. 1, p. 3–10. https://doi.org/10.2478/aei-2023-0001.

[27] Nielsen J. Usability Engineering. Morgan Kaufmann Publishers Incorporated. 1994.

Lisad

Lisa 1. Näidisprojektist genereeritud Exceli fail

Lisas 1 on välja toodud näide EbA-UIM rakenduses näidisprojektist genereeritud Exceli failist.

ID	Funktsionaalsuse kirjeldus		Sidusrühm	Funktsionaalsuse eeltingimus	Võrreldav situatsioon	Sama sidusrühm?	Sama kontekst?	Eesmärgipärane kasutus?	Lahendusega rahul?	Prioritiseeri	
32	After the customer has chosen a service (women's, men's, or children's hairdresser), the system asks for the hair length (short, shoulder-length, long).	Do	elderly customers	know why it is crucial to choose the right hair length.	Previously reporting the correct hair length.	Jah	Osaliselt	Osaliselt	Osaliselt		In a slightly differ precon
		Do	elderly customers	know what length is short, shoulder length, or long hair.	They must have known the hair length when booking over the phone.	Jah	Jah	Osaliselt	Osaliselt	1	There is somewhat
		Do	elderly customers	understand that they are being asked what their hair length is at the time of booking, not the desired size with a cut.	Customers who have previously made a reservation on the website have chosen their current hair length.	Osaliselt	Jah	Osaliselt	Osaliselt	2	There is somewhat dissa
		Do	elderly customers	know what length is short, shoulder length, or long hair.	Customers who have previously made a reservation on the website have had to know the hair length.	Osaliselt	Jah	Osaliselt	Osaliselt	2	There is somewhat dissa
		Do	elderly customers	understand that they are being asked what their hair length is at the time of booking, not the desired size with a cut.	When booking by phone, the hairdresser asks for the current length of the hair.	Jah	Jah	Osaliselt	Osaliselt	1	There is somewhat
33	After the customer has selected the hair length, the system displays the duration and price of the service, as well as the fields "Choose a hairdresser", and "Choose a date". 7.1. After the client has selected the hairdresser's name from the drop-down menu, the system displays a side-by-side calendar view, where the free dates of this hairdresser are displayed with a distinctive background color, and the rest of the dates are inactive. If the customer selects the name of another hairdresser, the system will make the free dates of the chosen hairdresser active. 7.2. When the customer selects by date, the system displays the available dates to the customer with a distinctive background tone. When the customer clicks on the date, the system displays the start time of the service. If the customer changes the date, the system will show the available hours for the selected date. Once the customer has chosen the start time of the service, the system displays the hairdresser's name and the button "Continue booking".	Do	elderly customers	understand they can make a choice based on the hairdresser's name or the desired date. However, these two criteria may not match.	When making a reservation by phone, customers can ask either by the name of the hairdresser or by the desired date. These two criteria may not match.	Jah	Jah	Jah	Osaliselt		There is s
		Do	elderly customers	remember the hairdresser's name if they want to choose a hairdresser by name.	When booking by phone, if customers want to book by the hairdresser's name, they remember the hairdresser's name.	Jah	Jah	Osaliselt	Osaliselt		There is somewhat
		Do	elderly customers	understand the basis on which the service duration and price are formed.	Understanding the basis when booking by phone, the customer is told how long the procedure will last and how much it will cost.	Jah	Jah	Jah	Jah	1	
		Do	elderly customers	understand that a distinct color on the calendar means that those dates are available at that hairdresser.	Allowing symbols, calendars, and similar infographics on the web that people use.	Osaliselt	Jah	Jah	Jah	1	
		Do	elderly customers	, after choosing the date, realize that they have to select only the service start time from the displayed times.	When booking by phone, the start time of the service is stated.	Jah	Osaliselt	Jah	Jah		
		Do	elderly customers	understand the basis on which the service duration and price are formed.	The duration and price of the service are displayed for those who have previously made a reservation on the website.	Osaliselt	Jah	Jah	Jah	2	
		Do	elderly customers	understand that a distinct color on the calendar means that those dates are available at that hairdresser.	Those who have previously made a reservation on the website have chosen the valid date, which is marked in a different color.	Osaliselt	Jah	Jah	Jah	1	
		Do	elderly customers	know that this is not the end of the reservation yet; they have to click the "Continue reservation" button and continue the activity.	When placing an order in the e-store, clients must press the button to continue the order.	Ei	Jah	Jah	Osaliselt	2	There is a diffe precondition, but ther
		Do	elderly customers	know that this is not the end of the reservation yet; they have to click the "Continue reservation" button and continue the activity.	Those who have previously made a reservation on the website have had to continue the reservation by clicking on the continue button.	Osaliselt	Jah	Jah	Osaliselt	1	There is so
34	After the customer has clicked "Continue booking", the system will show the customer a summary of the previously made booking and ask to fill in the fields: first name, last name, e-mail, phone, and "Confirm booking". The customer clicks "Confirm booking," but if any fields are not filled in, the system marks this place in red and displays the text: "Please fill out all required fields."	Do	elderly customers	know that it would be wise to check the summary of the previous reservation.	When booking by phone, the hairdresser repeats the booking summary. It's essential to listen to whether everything is as you wish.	Jah	Jah	Jah	Jah	1	
		Do	elderly customers	know that it would be wise to check the summary of the previous reservation.	Those who have previously made a reservation on the website should have checked the summary.	Osaliselt	Jah	Jah	Jah	2	
		Do	elderly customers	they know that it is possible and know how to go back to the previous view if some information is incorrect in the booking summary.	When booking by phone, the customer must react to the summary if something is wrong.	Jah	Osaliselt	Jah	Jah	1	
		Do	elderly customers	they know that it is possible and know how to go back to the previous view if some information is incorrect in the booking summary.	Customers who have previously made a reservation on the website have had to navigate back if something went wrong.	Osaliselt	Jah	Jah	Ei tea	2	S
		Do	elderly customers	know that at this stage, no data has been sent to the hair salon, and they can still make changes or cancel the booking.	When placing an order in the e-store, no money transfer has been made until confirmation.	Osaliselt	Jah	Jah	Jah		
		Do	elderly customers	understand why they are asked for their first and last names.	Other services also ask for information and require fields to be filled out, as well as non-online services.	Jah	Jah	Jah	Jah		
		Do	elderly customers	understand why they are asked for an email address or phone number.	When booking by phone, an e-mail address or phone number is asked.	Jah	Jah	Jah	Jah		
		Do	elderly customers	have email addresses or phone numbers they use regularly to get notifications or get notifications.	E-mail address or phone number they have been used regularly for previous hairdresser appointments.	Jah	Jah	Osaliselt	Jah		
		Do	elderly customers	understand that the email address or phone number they give must be the one they use regularly.	They give a phone number and e-mail address for other services, including non-online services.	Jah	Jah	Osaliselt	Jah		
		Do	elderly customers	have entered the correct contact data.	They are entering the data into other online services (e.g., e-shop)	Jah	Jah	Ei tea	Ei tea		The action outco
		Do	elderly customers	have entered the correct contact data.	Customers who have previously made a reservation on the website must have entered the correct contact data.	Osaliselt	Jah	Jah	Jah	2	
		Do	elderly customers	know that the "Confirm booking" button has to be pressed to complete the booking.	To complete a transaction in the e-bank, the customer must click the confirmation button.	Osaliselt	Jah	Jah	Jah		
		Do	elderly customers	know that the "please fill in all fields" message means that some required information has not been filled in and needs to be filled.	Required fields have also been used when filling out paper documents.	Jah	Osaliselt	Jah	Osaliselt		In a slightly differ
		Do	elderly customers	know that the "please fill in all fields" message means that some required information has not been filled in and needs to be filled.	Those who previously made a reservation on the website have had to react to the notification.	Osaliselt	Jah	Jah	Ei tea	1	S
35	After the customer has confirmed the reservation, the system displays the following notification to the customer: "Your appointment has been	Do	elderly customers	read the confirmation information that is displayed.	Reading the confirmation information displayed when completing the e-shop or online bank service.	Osaliselt	Jah	Ei tea	Ei tea	2	The action outco

Answering machine text | Online booking | Reminder

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Sander Põldma** ,
(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

„Kasutatavuse kogemuspõhise analüüsi meetodi“ rakenduse edasiarendus ja tiimitöö juhtimine ,
(lõputöö pealkiri)

mille juhendaja on **Anu Piirisild** ,
(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Sander Põldma

15.05.2025