TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Gerdo Germann

Testimised ja macOSile kohandamine „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele

Bakalaureusetöö (9 EAP)

Juhendaja:
Anu Piirisild, MSc

Tartu 2025

Testimised ja macOSile kohandamine „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele

Lühikokkuvõte:

Bakalaureusetöö eesmärk oli arendada edasi EbA meetodi töölaurakendust EbA-UIM, viies selle üle platvormist sõltumatule Electroni raamistikule. Arendus toimus neljaliikmelises tiimis, kus töö autor vastutas rakenduse toimivuse eest macOS ja Linuxi operatsioonisüsteemidel. Töö autor testis rakendust käsitestimise meetodiga ja viis läbi kasutatavuse testimise, rakendades valjult mõtlemise meetodit. Testide tulemused andsid väärtuslikku tagasisidet rakenduse täiustamiseks. Töö tulemusena valmis kolme platvormi ülene EbA-UIM rakendus, mis on varasema versiooniga võrreldes kasutajasõbralikum.

Võtmesõnad: EbA meetod, töölauarakendus, platvormist sõltumatu arendus, Electron raamistik, käsitestimine, kasutatavuse testimine

CERCS: P175 Informaatika, süsteemiteooria

„Experience-based Analysis method” desktop application testing and macOS optimisation

Abstract:

The aim of this thesis was to further develop the EbA method desktop application by migrating it to a cross-platform Electron framework. The author was responsible for ensuring the application's functionality on macOS and Linux. The author tested the application through manual testing methods and usability testing, using the think-aloud method. Test results provided valuable feedback for further improvements. As a result, in comparison to an earlier version, a more user-friendly, cross-platform EbA-UIM application was delivered.

Keywords: EbA method, desktop application, cross-platform development, Electron framework, manual testing, usability testing

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Kasutatavuse kogemuspõhine analüüs (ingl *Experience-based Analysis*) (edaspidi lühendatud EbA) kui meetod on loodud eesmärgiga muuta digitoodete ja e-teenuste analüüsi faas kuluefektiivsemaks [1]. EbA meetod aitab tootearenduse varases etapis saada aimu, kas lahendus kavandatud moel hakkab toimima ning millised on võimalikud riskid, seda ennekõike kasutajate käitumise ja harjumuste seisukohast [1].

EbA meetodi mugavamaks kasutamiseks on 2024. aastal valminud töölauarakenduse I etapp Windows operatsioonisüsteemile. Rakenduse I etapis loodud versioon sisaldab põhilisi funktsionaalsusi. Rakenduses on võimalik koostada tabelistruktuuris EbA meetodiga analüüs enda projektile. Rakenduse II etapi eesmärkideks on täiendada mõningaid olemasolevaid funktsionaalsuseid, lisada uusi funktsionaalsuseid ja elemente rakenduse tabelivaatesse, tõsta digiligipääsetavust, lahendada rakenduse aeglaseks muutumise probleem ning viia rakendus platvormist sõltumatuks (ingl *cross-platform*) ehk kasutamiseks ka macOS ja Linux arvutitel. Vältimaks olukordi, kus rakenduse välja arendamisel leitakse uus kriitiline viga, on soov rakendusega II etapis teha erinevaid testimisi. Eelnevalt kirjeldatud edasiarendamise protsessi nimetatakse edaspidi EbA meetodi töölaurakenduse II etapiks, lühendatult rakenduse II etapp. Kavandatud arendusprotsess sooritatakse neljaliikmelises tiimis.

Bakalaureusetöö peamine eesmärk on arendada edasi I etapi töölauarakendust ning läbi viia erinevaid testimisi. Edasiarendamise all peetakse silmas rakenduse üle viimist uuele, platvormist sõltumatule raamistikule. Riigikohus tõi 2025. aastal välja oma otsuses [2], et tänapäeval kavandatud tehnoloogilistel lahendustel peavad olema kõrged nõudmised ning need peavad olema kasutajasõbralikud. Niisuguse nõudmise täitmiseks on sobilik sooritada kasutatavuse testimine rakendusele.

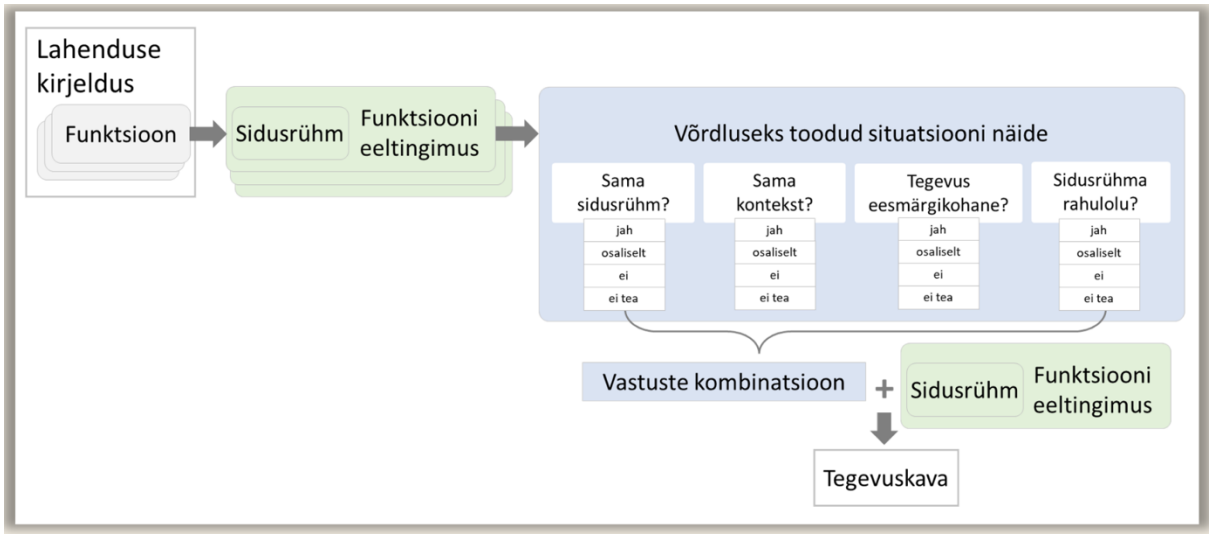
Bakalaureusetöö on jagatud neljaks peatükiks. Esimeses peatükis kirjeldatakse lähemalt EbA meetodit, rakenduse II etapi meeskonnatöö korraldust ning kirjeldatakse lähemalt töö autori ülesandeid. Lisaks kajastatakse teoreetilisi põhimõtteid töölauarakenduse edasiarendamisele ning testimistele. Töö teises peatükis kirjeldatakse autori panust rakenduse edasiarenduses ning selgitatakse macOS ja Linuxi eripärasid arendamisel. Töö kolmandas peatükis on kirjeldatud rakendusega sooritatud testimisi. Neljandas, töö viimases peatükis on välja toodud töö tulemused, väljakutsed ning edasiarenduse plaanid.

1. Taust

Töö esimeses peatükis kirjeldatakse EbA meetodit, rakenduse I etapi seisu ülevõtmise hetkel, rakenduse II etapi eesmärke ja meeskonnatöö korraldust. Lisaks kirjeldatakse arendusega seotud tarkvara ning testimismeetodeid.

1.1 EbA meetod

Järgmine lõik põhineb Anu Piirisilla jt [1] artiklil ning EbA meetodi juhendil[1], mis on kättesaadav ka EbA meetodi töölauarakendusest. EbA meetod on loodud eesmärgiga saada tootearenduse varases etapis aimu, kas lahendus kavandatud moel hakkab toimima ning millised on võimalikud riskid, seda ennekõike kasutajate käitumise ja harjumuste seisukohast. EbA meetodis tehakse järeldusi kasutatavuse kohta vaadates neid aspekte, mida sidusrühmadelt eeldatakse, et loodav toode toimiks ettenähtud viisil ning vastavalt kuidas sidusrühmad on varasemalt mõnes muus situatsioonis neid eeltingimusi täitnud. Lisaks hinnatakse, milline oli seejuures nende rahulolu. EbA aitab kaardistada, mil määral on sidusrühmadel olemas neilt eeldatud teadmised, oskused või vahendid, mis on toote edukaks toimimiseks vajalikud. Joonisel 1 on visuaalselt kujutatud, kuidas EbA meetodit analüüsiprotsessis rakendatakse.



Joonis 1. EbA meetodi raamistik. Joonise autor: Anu Piirisild[1].

EbA meetodi mugavamaks kasutamiseks on loodud töölauarakendus, mida nimetakse EbA-UIM rakenduseks.

[1] EbA-UIM rakenduse II etapi rakenduse repositoorium: https://bitbucket.org/eba-method/eba-code/src/master/

1.2 Töövahendi I ja II etapp

EbA meetodi töövahendi arendamine on jagatud mitmesse etappi. Rakenduse I etapp valmis õppeaastal 2023/2024 Tartu Ülikoolis bakalaureuse tudengite Karl Olaf Kuldmaa [3] ja Iris Kreinini [4] meeskonnatöö tulemusel koostöös EbA meetodi autori Anu Piirisillaga. Töö I etapi tulemusena valminud versioon sisaldab peamisi funktsionaalsuseid:

1) rakenduses saab kasutaja omaenda projekti analüüsida;
2) saab kasutada eesti ja inglise keeles;
3) rakendus on alla laetav Windows arvutitele ning sisaldab näidisprojekti eesti ja inglise keeles [5].

Töölauarakenduse arendamise II etapi üheks peamiseks eesmärgiks on muuta rakendus platvormist sõltumatuks (ingl *cross-platform*), et see oleks kasutatav lisaks Windowsile ka macOS ja Linux operatsioonisüsteemidel. Samuti on selles arendusfaasis plaanis täiendada olemasolevaid funktsionaalsuseid, lisada uusi funktsioone ja elemente rakenduse tabelivaatesse, parandada digitaalse ligipääsetavuse taset ning lahendada probleeme, mis põhjustavad rakenduse kiiruse langust.

II etapp valmib koostöös nelja tudengiga – Sander Põldma [6], Rahel Pettai [7], Timo Kaasik ja siinse töö autor. Tudengitele on teise etapi arendusprotsess bakalaureuse lõputööks. Rakenduse II etapi arenduse töökorraldust kirjeldab järgnev alapeatükk.

1.3 Tiimisisene töökorraldus

Käesolev alapeatükk kirjeldab rakenduse teise etapi töökorraldust. Rakenduse II etapi arenduse tööülesanded olid jaotatud tudengite vahel vastavalt igaühe uurimisteemale. Sander Põldma üheks ülesandeks oli arendada edasi nii rakenduse ees- kui ka tagasüsteemi ning viia töölauarakendus üle uuele platvormist sõltumatule raamistikule (edaspidi kompleksarendaja). Lisaks nendele ülesannetele oli tema roll juhtida tiimitööd ning koordineerida suhtlust. Sellises kontekstis nimetatakse edaspidi Sander Põldmad tiimijuhiks. Rahel Pettai tegeles rakenduse eessüsteemi arendamisega (edaspidi eessüsteemi arendaja) ja digiligipääasetavuse nõuetega (edaspidi digiligipääsetavuse disainer). Timo Kaasiku ülesandeks oli teha II etappi valitud nõuetele analüüs ning jätkata nõuete arendusega rakenduse eessüsteemis. Rakenduse II etapi arendusse oli kaasatud ka EbA meetodi looja Anu Piirisild (edaspidi tooteomanik).

Siinse töö autori tööülesanneteks on:

1) töölauarakenduse üleviimine uuele platvormile, kontrollides, et see oleks platvormist sõltumatu ja kohanduks macOSil, vajadusel ka Linuxil;
2) rakenduse II etapi arenduste testimine ning automatiseeritud testimise arendamine töölauarakendusele;
3) läbi viia kasutatavuse testimine lõppkasutajatega.

Alljärgnevates peatükkides viidatakse ka teiste tudengite töödele, juhul, kui kaastudeng oli ülesande sooritamisse kaasatud või kui täpsem selgitus kajastub vastava tudengi bakalaureuse lõputöös.

1.4 Platvormist sõltumatud raamistikud töölauarakendustele

Rakenduse II etapis on planeeritud viia EbA-UIM rakendus platvormist sõltumatuks. Arendamise I etapis valminud töölauarakendus töötas Windows operatsioonisüsteemiga arvutitel. Kuna see katab ainult osa kasutajate keskkondadest, siis II etapi üheks eesmärgiks sai püstitatud platvormist sõltumatu töölauarakenduse ehitamine. See tähendab, et olemasolev kood tuleb üle viia teistsugusele raamistikule ja ülesehitusele kui I etapi rakendus, kuna I etapi programmikood ei toeta teisi operatsioonisüsteeme. Käesoleva lõputöö üks eesmärkidest on veenduda, et II etapi uus raamistik toetaks lisaks Windowsile ka macOSi. Lisaks raamistiku valimisele tuleb katsetada erisusi macOS keskkonnaga. Uue töölauarakenduse ülesehitus peab samuti olema töökindlam ja kiirem kui eelmine versioon. Viimati nimetatud töövaldkonda käsitleb kompleksarendaja lõputöö [6].

Platvormist sõltumatuid raamistikke kaardistades leiab mitmeid vasteid. Peamised kolm raamistikku, millele erinevates netifoorumites viidatakse, on Electron[2], Flutter[3] ja .NET[4] raamistik. Electroni koduleht selgitab, et Electroni raamistik võimaldab arendada platvormist sõltumatut töölauarakendust nii Windows, macOS kui ka Linux tarkvarale. Põhilisteks Electronis kasutatud programmikeelteks on JavaScript, HTML ja CSS. Flutter on Google'i raamistik. Flutteri suurim erinevus võrreldes Electroniga on see, et koodi kirjutatakse omandkeeles Dart. .NET on Microsofti raamistik, mis samuti kasutab omandkeelt, kuid ka lisaks C#'i. Tasub mainida, et kõik eelnevad raamistikud on lähtekoodtarkvarad ja lisatasuta kasutamiseks nii kommerts kui ka isiklikuks tarbeks.

[2] https://www.electronjs.org
[3] https://flutter.dev
[4] https://dotnet.microsoft.com/en-us/

Elias Müller võrdles oma bakalaureusetöös [8] Electroni sarnaselt toimiva raamistiku Flutteriga. Ta leidis, et arendades sama süsteemi Flutteriga, saavutati 30% vähem koodiridu, aga ei saadud öelda, et kood oleks kiirem või kasutaks vähem mälu. Niisamuti oli Mülleril uue projekti ülesseadmine Flutteriga tülikam, kuna nõudis Flutteri omakeele Darti oskusi ja teatud eripärasid. Mülleri töö puhul on oluline ka mainida, et kasutati Flutteri beetaversiooni.

Sarnasel kombel on 2019. aastal võrreldud Electroni ja .NET raamistiku Daniyar Alymkulovi bakalaureusetöös [9], kus arendatav rakendus valmis ainult Electroni toel. Nagu ka eelneva tööga, leidis Alymkulov, et Electron on parim valik töölauarakenduse arendamiseks. Seda põhjusel, et .NET raamistik võib olla rohkem Windowsi süsteemile arhitektuuriliselt lähedasem kui teistele operatsioonisüsteemidele ja seega võib põhjustada teadmatuid probleeme tarkvara platvormist sõltumatu rakenduse valmistamiseks.

Rakenduse II etapis sai platvormist sõltumatu töölauarakenduse arendamise raamistikuks valitud Electroni tarkvara. Valik langes Electroni kasuks, kuna I etapi programmikoodis [5] on palju kasutatud Angular veebiraamistikku. Kuna Electron toetab Angulari ka ilma täiendavate mooduliteta, on pärast I etapi arendust võimalik rakendus uuele raamistikule üle viia ilma ulatuslike muudatuste ja lisakomplekssuseta. Olles platvormist sõltumatu raamistik, sobib Electron hästi nii macOSi kui ka Linuxi töölauarakenduse arendamiseks.

1.5 Rakenduse testimise meetodid

Veendumaks arendusprotsessi ja selle väljundi toimimises ning vastavuses nõuetega, on vajalik testida tulemusi. Autor selgitab metoodikaid nii käsitestimise, automatiseeritud kui ka kasutajatele suunatud testimiste kategooriatest. Järgnevad alapeatükid tutvustavad testimismeetodeid, lahendusi ning võimalikke tarkvara raamistikke.

1.5.1 Käsitestimine

Käsitestimist defineeritakse kui „testimist, mille sooritavad inimesed sisestades teavet testimisobjekti ja verifitseerides tulemeid; vastupidiselt automatiseeritud testimisele ei kasuta instrumente, roboteid ega muid testimismootoreid“[5].

Järgnev lõik tugineb Juha Itkonen jt [10] artiklile, kus nad kirjeldavad käsitestimise eri tüüpe. Käsitestimisel eksisteerib erinevaid alamtüüpe – nii selliseid, kus koostatakse detailsed testimisplaanid, aga ka selliseid, kus testimise vorm on vabam ehk vabatestimised.

[5] https://akit.cyber.ee/

Käsitestimiste erinevate alamtüüpide uurimiseks analüüsisid Juha Itkonen jt 11 testimissessiooni, mille käigus paluti tarkvara arendajatel, kelle üks tööülesanne oli tarkvara testimine, sooritada oma tavapäraseid tegevusi. Leiti, et paljud testimise vormid, mida kasutati vabatestimisel, sisaldasid enimlevinud tavasid ja kogemusest saadud teadmisi, kuidas tuvastada vigu.

Üks vabatestimise alamliike on suvatestimine (ingl a*d-hoc testing*). Suvatestimise kirjeldus on refereeritud Chris Agruss ja Bob Johnsoni [11] artiklist. Agruss ja Johnson kirjeldavad, et suvatestimisel tehakse testimine minimaalse dokumenteerimisega; uuritakse ainult üks kord, kui just ei avastata viga. Agruss ja Johnson rõhutavad, et tavapärasel testimisel, enne rakenduse uue versiooni välja laskmist, on siiski oluline dokumenteerida testimise leiud ning kuidas kindlaid teste sooritati. Kuna suvatestimine oma loomult on väga juhuslik ja vaba oma vormilt, aitab see läheneda testimisobjektile loominguliselt. Loominguline lähenemine võimaldab avastada vigu, mida tavapärases, kitsalt määratud vormis ei pruugi leida. Suvatestimist sobib kasutada igal ajal kui arendatakse tarkvara ning see aitab laiendada katsetaja teadmist testitavast süsteemist.

Imran Bhatti jt [12] kirjeldavad suvatestimise kõrval ka teisi käsitestimise alamtüüpe:

1. Regressioontestimine (ingl *regressioon testing*) on testimise tehnika, kus veendutakse, et süsteem töötab endiselt korrektselt pärast muudatuse tegemist koodis või pärast uue mooduli lisamist süsteemi.
2. Vastuvõtutestimine (ingl *acceptance testing*) on testimise meetod, kus lõppkasutajad testivad tarkvara vastavust nõuetele.
3. Proovitestimine (ingl *smoke testing*) on testimise meetod, mida kasutatakse tarkvara kõige olulisemate funktsioonide kontrollimiseks, et veenduda nende töökorras olekus ja selles, et põhiline funktsionaalsus on säilinud.

Imran Bhatti jt [12] rõhutavad, et suvatestimisel on oluline esmalt mõista ja omada piisavalt teadmisi testitava tarkvara kohta. Testimiseks tuleks valida sobivad meetodid, vajadusel kasutada ka hübriidset lähenemist, ning kõik leitud vead tuleb korralikult dokumenteerida. Kuna suvatestimine ei järgi kindlat vormi ja on ajaliselt vähekoormav, sobib see hästi agiilsesse arendustsüklisse, kuna see võimaldab arendajatel kiiresti saada tagasisidet tehtud muudatuste ja võimalike vigade kohta.

1.5.2 Automatiseeritud testimine

Vahid Garousi ja Mika Mäntylä [13] nimetavad automatiseeritud testimiseks tarkvara testimise lahenduste automatiseerimist, mis tähendab, et kasutatakse spetsiaalset tarkvara kontrollimaks testide jooksutamist ja nende tulemusi.

Vahid Garousi ja Mika Mäntylä [13] toovad esile, et enne kui valitakse automatiseeritud lahendus, on oluline veenduda testiva süsteemi küpsusest. Alati ei ole mõistlik automatiseerida, näiteks kui lahendus ei ole lõplik või esineb pidevaid muutusi. Artiklis märgitakse, et pidevalt muutuvate rakenduste puhul võivad automatiseerimise eelised kaduda, sest iga muudatuse järel tuleb automatiseeritud teste uuesti kohandada. Sellises olukorras võib olla mõistlikum kasutada manuaalset testimist, eriti kui süsteem ei ole veel arenduse lõppjärgus.

Automatiseeritud testimise valikul on hea teada mõningaid aspekte, mille on välja toonud Elis Pelivani ja Betim Cico oma artiklis [14], millele ka järgnev lõik tugineb. Enne veebirakenduse testimise automatiseerimist peab olema teadlik rakenduse äriloogikast. Äriloogikale baseerub automatiseeritud testide struktuur ning kuidas ja mis järjekorras antud teste jooksutatakse. Testid on kirjutatud vastavalt valitavale tarkvarale, mis lubab testi automatiseerida. Pelivani ja Cico on veel täheldanud, et automatiseerimise protsessis on tähtis valida raamistik, mida on võimalikult kiiresti ja kergelt kasutusele võtta, et tarkvara kataks nõutud testimise vajadusi. Pelivani ja Cico lisavad veel, et hea oleks kui ka tarkvara võimaldab koostada testide kohta kvantitatiivset raportit.

Electroni kodulehel[6] on veebiliidese testide automatiseerimisel Electroniga viidatud kolmele testimise raamistikule. Nendeks on WebDriverIO[7], Selenium[8] ja Playwright[9]. Kõigi kolme puhul on ühendamisel Electroniga võimalik lahendada läbivtestimist (ingl *end-to-end testing*) kasutajaliidesel. Tarkvarade võrdluses esineb erinevusi seadistamisel, kuid soorituse (ingl *performance*) kohta pole Electroni koduleht informatsiooni jaganud.

1.5.3 Kasutatavuse testimine

Kasutatavuse testimise läbi viimine aitab aru saada, kuidas projekti antud hetkel kasutatakse ning mida teha selle parandamiseks [15]. Kate Morani [16] järgi on kasutatavuse testimise läbiviimiseks vajalik kolm põhielementi: läbiviija, ülesanded ja osalejad. Läbiviija ülesanne on

[6] https://www.electronjs.org/docs/latest/tutorial/automated-testing#using-the-webdriver-interface
[7] https://webdriver.io
[8] https://www.selenium.dev
[9] https://playwright.dev

suunata osalejaid läbi testimisprotsessi, tagades selle sujuva kulgemise. Osalejateks on sihtrühma kuuluvad isikud, kelleks on rakenduse eeldatavad lõppkasutajad. Ülesanded tähistavad tegevusi, mida kasutajad testimise käigus tootega tegelikult sooritavad.

Raluca Budiu [17] toob välja, et kasutatavuse testimist on võimalik läbi viia nii kvantitatiivsel kui ka kvalitatiivsel viisil. Kvantitatiivne lähenemine keskendub mõõdetavatele näitajatele, nagu näiteks see, kui kiiresti kasutajad suudavad konkreetseid ülesandeid täita või millisel määral nad suudavad eesmärgid edukalt saavutada. See lähenemine võimaldab koguda andmeid, mida saab statistiliselt analüüsida. Budiu sõnul võimaldab aga kvalitatiivne testimine sügavamalt mõista kasutajakogemust, kuna see toob esile kasutajate mõtlemisprotsessid, segadust tekitavad disainielemendid ning ootamatud kasutamismustrid, mis võivad jääda kvantitatiivses analüüsis märkamata.

See, kui palju inimesi osaleb ühel testimise sessioonil, sõltub paljuski sellest missugust testimise meetodit kasutatakse, kas kvantitatiivset või kvalitatiivset [16]. Järgnev lõik on refereeritud Nielseni [18] artiklist. Kvantitatiivsel lähenemisel, kust soovitakse saada mõõdetavaid tulemusi, on hea tava kaasata vähemalt 20 inimest. Kvalitatiivsel lähenemisel, kus testitakse disaini, piisab üldiselt viiest inimesest, väiksemate projekti puhul vähemalt kahest. Selleks uuris Nielsen 83 kasutatavuse testimise konsultatsiooni projekti. Uuringust selgus, et väiksemate osalejate arvu korral tuvastati rohkem kasutatavusest tulenevaid leide.

Lõppkasutajatega testimisel peetakse oluliseks, et läbiviija saab adekvaatset tagasisidet lõppkasutajalt [16]. Selle jaoks rakendatakse peamise meetodina kasutatavuse testimisel „valjult mõtlemise meetodit“ (ingl *Think Aloud Method*) [19]. Valjult mõtlemise meetod tähendab, et osaleja esitab oma mõtteid valjuhäälselt ning läbiviija kuulab ja salvestab sama mõtte [20]. Eksisteerib ka teisi meetodeid osalejate uurimiseks, nagu näiteks pilgujälitust (ingl *eye tracking*), kus osalejate silmade liigutusi jälgitakse spetsiaalse tarkvaraga [21].

Kasutatavuse testimisi on võimalik läbi viia nii kohapeal üheses ruumis, kui ka interneti kaudu videokõnega [22]. Kate Morani [16] järgi saab kasutatavuse testimist läbi viia kas kohapeal või distantsilt, kusjuures iga lähenemine jaguneb omakorda modereeritud ja modereerimata testimiseks. Kohapealse testimise puhul on tavapärane modereeritud vorm, kus testimist juhib vahetult kohal viibiv läbiviija. Morani sõnul võib distantsilt testimine toimuda nii modereeritult kui ka modereerimata. Viimase puhul annab osaleja tagasisidet kas spetsiaalse vormi kaudu või kasutatakse tarkvara, mis salvestab ja analüüsib testimiskäitumist ilma läbiviija otsese sekkumiseta.

James Lewis [21] on toonud välja, et põhitestimisele eelneb tavaliselt piloottestimise sessioon [21]. Tema sõnul näeb see ette meetodite kontrollimist, plaanide ja vormide valideerimist, et päris testimisel ei peaks organisatoorsete muredega tegelema.

Enne osalejatega testi läbiviimist on vajalik osalejatega sõlmida nõusoleku vorm [22]. Andmekaitse Inspektsiooni (AI) kodulehel [23] selgitatakse, et isiku nõusolekut tuleb küsida juhul, kui isikuandmete töötlemise eesmärk vastab Euroopa Liidu isikuandmete kaitse üldmääruses[10] sätestatule. AI rõhutab, et nõusolek peab olema andmesubjekti poolt antud vabatahtlik, teadlik, konkreetne ja ühemõtteline tahteavaldus. See võib toimuda kirjaliku avalduse või muu selgelt nõusolekut väljendava tegevuse kaudu. Nõusoleku tekst peab olema selgesõnaline, arusaadav ning kergesti kättesaadav. Lisaks peab olema täpselt määratletud, kes on andmete töötleja ja millisel eesmärgil andmeid töödeldakse. Samuti tuleb andmesubjekti teavitada tema õigusest nõusolek igal ajal tagasi võtta.

Testimisülesannete koostamisel peab lähtuma sellest, et need oleksid piisavalt arusaadavad, aga ka huvitavad osalejale lahendamiseks, et saada adekvaatset tagasisidet [24]. Ülesande vormistus peab ühtima uurimisküsimustega, milleks kasutatavuse testimist sooritatakse [21]. Dokumenteerimisel peab ära selgitama, mis on ülesande eesmärk, eeltingimused, loodetav tulemus ning ebaõnnestumise kriteerium [22]. Peale ülesannete sooritamist on heaks tavaks lasta osalejatel täita rahulolu küsimustik, mida saab analüüsida kvantitatiivsel meetodil [22].

Testimise edukaks sooritamiseks on oluline kutsuda testimisele rakenduse lõppkasutajad [21]. Lõppkasutajate kirjeldamisel on hea välja tuua tunnused, mis olid antud testimisel vajalikud [22]. Nagu näiteks osaleja töövaldkond, vanus, rakenduse kasutamise oskustase ning muud tunnused [21]. Oluline on ka jääda testimisel osalejate suhtes erapoolikuks, et vältida testitulemuste mõjutamist [24].

Tulemuste esitamiseks on omad metoodikad, nii kvantitatiivsel kui ka kvalitatiivsel kasutatavuse testimisel. Peamisteks esitamise viisideks loetakse summeeritud andmeid, s.t, et andmed kas üldistatakse, leitakse ühised omadused või loendavate tulemustega esitatakse statistikat [22]. Kvalitatiivsete tulemuste esitamisel öeldakse, et tuleb esitada hästi läbi mõeldud soovitusi [21]. Tulemusi on võimalik edastada nii tekstikujul kui graafilisel kujul, näitlikustamaks, mis on testimise leiud [22].

[10] https://ec.europa.eu/newsroom/article29/items/623051

2. Rakenduse macOSile kohandamine ja töölauarakenduse arendus

Järgnevad alapeatükid kirjeldavad käesoleva töö kontekstis lahendatud töölaurakenduse arendust uuel töölauarakenduse raamistikul. Täpsemini selgitakse macOSis selgunud omapärasusi ning kirjeldatakse macOSi ja Linux operatsioonisüsteemi rakenduse pakendamist.

2.1 Electroni algseadistus

Lähtuvalt eelnevale teadmistele, mida oli jagatud siinse töö teoreetilises peatükis 1.4, sai uueks töölauarakenduse raamistikuks valitud Electron. Electroni paigaldamiseks sai järgitud Electroni ametliku juhist nende kodulehelt[11] ja Israel Quiroz[12] juhendamist samateemalisest videost. Quiroze suunamine oli vajalik, kuna ametlikul juhendil polnud lisatud täpsustusi, kuidas Electroni ühendada Angular raamistikuga. Peale Electroni paigaldamist tuli veenduda, et ees- ja tagasüsteem toimivad. Toimivuse kontrollimisega tegeles kompleksarendaja, kes kajastab seda osa enda töös [6]. Käesoleva töö autori ülesanne oli veenduda, et lahendus toimiks ka macOS keskkonnas.

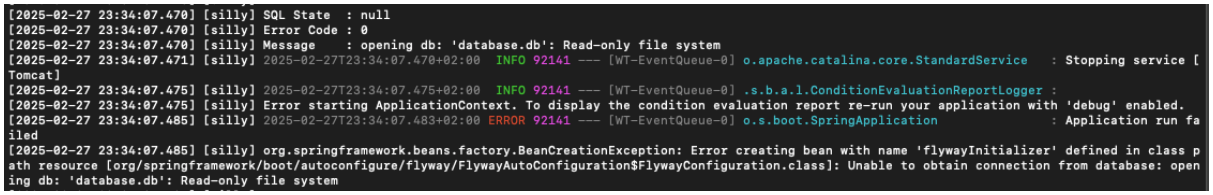
2.2 Electroni kohandamine macOSile

Electroni raamistikule üleviimisel on sõltuvalt operatsioonisüsteemist eripärasusi. Kuna rakenduse pakendamine ja turvanõuded käivitamisel varieeruvad platvormiti, tuli teha rakenduses kohandusi, et tagada selle sujuv ja platvormist sõltumatu toimimine kõigil operatsioonisüsteemidel.

2.2.1 Rakenduse tagasüsteemi tõrked macOSil

Kuigi esialgsed testimised peale Electronile üleviimist näitasid, et rakendus jookseb nüüd ka macOS süsteemil, siis Electroni rakenduse pakendamisel ei vastanud see enam vastuvõtukriteeriumitele. Rakenduse eessüsteem ei olnud täielikult kuvatud, s.t et rakendus ei töötanud korrektselt. Sellise probleemi lahendamine osutus väljakutseks, kuna Windowsi süsteemil sarnast viga ei tekkinud. Tutvudes rakenduse tagasüsteemi logidega selgus, et probleem seisnes failiõigustes. Joonisel 2 on võimalik tutvuda logifaili sisuga. Tagasüsteemi andmebaasi fail oli lukustatud olekus ehk ainult lugemiseks mõeldud.

[11] https://www.electronjs.org/docs/latest/tutorial/tutorial-prerequisites
[12] https://www.youtube.com/watch?v=Ckn5rH7q4P8



Joonis 2. Andmebaasi faili veateade.

StackOverflow's[13] leidus artikkel, mis kirjeldas tekkinud probleemi. Nimelt üritas rakendus luua andmebaasi faili otse rakenduse jooksutatavasse kausta, kuid selline lähenemine ei sobi operatsioonisüsteemile. Kiire lahendus oli suunata andmebaasi faili loomine mujale kausta, siinpuhul kasutaja „Dokumentide" kausta, ning rakendus töötas uuesti samaväärselt Windows'i operatsioonisüsteemiga.

Ilmnes, et faili loomine enda jooksustavas kaustas on vastuolus operatsioonisüsteemi turvaseadetega, mida nii lihtsalt ei ole võimalik välja lülitada ja pole ka tavakasutaja jaoks jätkusuutlik valik[14]. Eelneva foorumi artikli autor on välja toonud erinevaid failisüsteemide ligipääse, mis eksisteerivad macOS tarkvaral. Siinse probleemi kontekstis said oluliseks artikli sektsioonid „App Sandbox" ja „Mandatory Access Control (lühendatult MAC)", kuna eelmainitud probleemid on spetsiifilised macOSile. Rakenduse aediku (ingl *sandbox*) turvapääsmed lubavad jooksutataval rakendusel kasutada ressursse enda aediku alal, s.t piiratud failisüsteemis. MAC on kohustuslik turvanõue, mille läbi liiguvad kõik arvuti käsud. Kui teatud rakendus äratab kahtlust, ei lubata rakendusele kõrgemaid ligipääse kui vaja. Seega, läbi komplitseeritud loogika, suunates rakenduse avama/looma andmebaasi rakendust mujale kui jooksutavasse kausta, sai lahendatud rakenduse tagasüsteemi käivitamine macOSil.

2.2.2 Electroni rakenduse pakendamine

Electroni rakenduse pakendamisel tugineti samamoodi nagu seadistamisel Electroni ametlikule kodulehele[15]. MacOSi ja Linuxi puhul osutus valituks Electroni pakendamise tööriist electron-packager[16]. Installeri ehitamisel kasutatakse macOSil electron-installer-dmg[17] ja Linuxil

[13] https://stackoverflow.com/questions/55699157/packaged-electron-app-does-not-launch-when-run-from-dock-but-works-fine-when-run
[14] https://developer.apple.com/forums/thread/678819
[15] https://www.electronjs.org/docs/latest/tutorial/tutorial-packaging
[16] https://www.npmjs.com/package/electron-packager
[17] https://github.com/electron-userland/electron-installer-dmg

electron-installer-debian[18]. MacOSi rakenduse ehitamise kood on esitatud joonisel 3. Linuxi rakenduse komplekteerimise käsk on näidatud joonisel 4.

```
"package-mac-arm64": "electron-packager . EbA-UIM --overwrite --platform=darwin --arch=arm64 --icon=./icons/eba.icns --out=../release-builds",
    "dmg-arm64": "electron-installer-dmg ../release-builds/EbA-UIM-darwin-arm64/EbA-UIM.app EbA-UIM --icon=./icons/eba.icns --background=image.png --overwrite --out=../release-builds/macOS"
```

Joonis 3. MacOSi rakenduse pakendamise käsud.

```
"package-linux-arm64": "electron-packager . EbA-UIM --overwrite --platform=linux --arch=arm64 --icon=./icons/eba.ico --prune=true --out=../release-builds",
    "debian-installer-arm64": "electron-installer-debian --src ../release-builds/EbA-UIM-linux-arm64 --dest ../release-builds/linux-deb --arch arm64"
```

Joonis 4. Linuxi rakenduse pakendamise käsud.

Mõlema operatsioonisüsteemi rakenduse ehitamisel on jälgitud, et rakendusel säiliks platvormist sõltumatuse nõue. Selle tagamiseks on rakendus pakendatud macOSil kahele kiibi arhitektuurilisele platvormile, milleks on arm64 ning x64. Linuxil eksisteerib lisaks x64 versioonile ka arm64 versioon. Windows'i rakenduse pakendamist kajastab kompleksarendaja [6] enda lõputöös. Windows'i rakenduse pakendamisel on kasutatud Electroni raamistiku tööriista electron-forge[19]. MacOSi ja Linuxi puhul seda kasutusele ei võetud, kuna testimisel selgus, et töölauarakendus ei avane, kui pakendada koos forge'iga. Selle juurpõhjust ei suutnud töö autor limiteeritud aja jooksul tuvastada, kuid üheks põhjuseks võib lugeda rakenduse aediku turvanõudeid.

Rakenduse tagasüsteemi jooksutamiseks on pakendatud vastavalt platvormile kaasa ka Java programmeerimiskeel. Java kaasa pakendamine on vajalik eelkõige, kuna rakenduse

[18] https://github.com/electron-userland/electron-installer-debian
[19] https://github.com/electron/forge

jooksutamisel ei pruugi lõppkasutajal olla Java installeeritud arvutisse, kuid ilma Javata rakendus ei käivitu. Igale versioonile rakendusest on kaasa pakendatud rakenduse arhitektuurile vastav versioon Eclipse Temurin Javast[20]. Käesoleva töö raames on kokku pakendatud järgnevad versioonid rakendusest:

- macOS Apple Silicon
- macOS Intel-based processor
- Linux Debian x64
- Linux Debian arm64

Rakenduse pakendamiseks leiab Electroni kodulehelt soovituse see protsess automatiseerida[21]. Automatiseerimine tuleb abiks, kui Electroni rakenduse ehitamisel pole parasjagu ligipääsu kõikide operatsioonisüsteemide seadmetele. Electroni rakenduse pakendamisel või ehitamisel on näiteks macOS platvormil peal piirang, kus rakenduse ehitamiseks on vajalik macOSil põhinev seade. Selle jaoks otsustas töö autor ehitada pidevintegratsiooni ja pidevvalmiduse konveieri[22] (ingl *CI/CD pipeline*).

Konveieri jaoks valmis YAML-fail, mis paigutati koodi repositooriumi juurkausta. Konveieri tegevuste ettevalmistamisel lähtuti samadest põhimõtetest nagu käsitsi ehitamisel:

1. Konveier jooksutab ennast, kui uuendused on suunatud „master“ harule.
2. Esialgu kompileerida rakenduse tagasüsteem, vajadusel jooksutada teste.
3. Kompileerida rakenduse eessüsteem, vajadusel jooksutada teste.
4. Pakendada rakendus koos electron-packager tööriistaga.
5. Ehitada rakenduse installer koos vajalike tööriistadega.

Lisaks nendele nõuetele peaks konveieri lahendus olema lisatasuta ning ühilduma EbA-UIM rakenduse II etapi repositooriumiga Bitbucket.

Konveieri koostamisel esinesid aga asjaolud, mis takistasid edasist arendust. Esiteks oli Bitbucketi repositooriumis keeruline kokku pakendada macOSi rakendust, kuna otsest lahendust electron-installer-dmg näol ei eksisteerinud. Eelmainitud tööriist töötab ainult macOS süsteemil. Samuti ka Windows süsteemile ei leitud lahendust. Rakenduse ehitamiseks

[20] https://adoptium.net/temurin/releases/
[21] https://www.electronforge.io/core-concepts/build-lifecycle#cross-platform-build-systems
[22] https://www.geeksforgeeks.org/what-is-ci-cd/

vajalikud tööriistad on samuti saadaval ainult Windows operatsioonsüsteemil. Bitbucketi konveier jookseb Linuxil põhineval tarkvaral.

Seega on loodud konveieri loogika oluliselt lihtsustatud kui eelnevalt kirjeldatud. Programmikoodi repositooriumis saab töö esitamisel automaatselt ehitada Linux Debianil töötava töölauarakenduse ja lisada selle Bitbucketi allalaadimiste kausta. See annab tooteomanikule võimaluse vajadusel rakendus ka Linuxi jaoks pakendada mugavalt.

3. Rakenduse testimine

Peatükis kirjeldatakse siinse töö autori panust töölauarakenduse arenduskäigus esinenud testimistel. Selgitatakse käsitestimise metoodikat, automatiseeritud testimise vajalikkust ning kasutatavuse testimist lõppkasutajatega.

3.1 Käsitestimise läbiviimine

EbA meetodi töölauarakenduse arendusprotsessi käigus on käesolevas töös kasutusele võetud hübriidlahendus suvatestimisest ja regressioontestimisest. Hübriidlahenduse kasuks ajendas otsustama testimise kiirus, minimaalse dokumentatsiooni koostamise vajadus ning EbA meetodi töölauarakenduse eripärad. Eripärade all on silmas peetud rakenduse erilist tabelilahendust ning komponentide omavahelist koostöötamist. Välja töötatud töölauarakenduse lahendus ning arendusprotsess vajavad kiiret ning tõhusat testimise meetodit.

Käsitestimine algas arendusprotsessis tekkinud muudatuse analüüsimisel, ehk peale muudatuse tõmbekutse heaks kiitu „dev“ harule. Igal rakenduses tehtud muudatusel eksisteeris oma Jira pilet, kus oli kirjeldatud tehtud tööd. Kirjelduse või piltide või mõlema põhjal alustati käsitestimist töölaurakenduses. Igale testimisele eelnes uue töölauarakenduse pakendamine, mis omakorda võeti aluseks muudatuse testimiseks.

Käsitestimisel olid töö autoril testijana valitud peamisteks vastuvõtukriteeriumiteks järgmised tingimused:

1. Kavandatav muudatus ei tee rakendust katki, s.t et rakenduse teised osad töötavad samaväärselt edasi.
2. Muudatus peab olema kuvatud mõistlikult, s.t et muudetud osad ei ole häirivad kasutajakogemusele, ning töötavad vastavalt kirjeldusele.
3. Muudatus ei sega platvormist sõltumatu töölauarakenduse arendamist, s.t et muudatus peab eksisteerima nii Windows, macOS kui ka Linux seadmetel.

Testimise käigus tuvastatud programmivead (ingl *bug*), raporteeriti Jira piletisse ning suunati tagasi arendajale. Testimise käigus raporteid või aruandeid ei koostatud ning see ei olnud ka testija töökavas. Kui kavatsetud muudatus läbis testimise vastuvõtukriteeriumeid, siis liigendati see Jiras vastavalt lahendatuks. Kui mitte, siis jällegi nii nagu eelnevalt mainitud, kommenteeriti viga Jira keskkonnas.

3.2 Automatiseeritud testimise lahendus

Käesolevas töös on soovitud automatiseerida mõningaid komponentide testimisi ja läbivtestimist, et lihtsustada arendaja töövoogu ning kiirendada arendamise ning testimise protsessi. Automatiseerimiseks valiti siinses töös hübriidvariant, see tähendab, et kokkuleppeliselt automatiseeritakse rakenduse põhifunktsioonid ning manuaalselt on testitud nõudeid, mis on vähemtähtsad. Kasutatavuse testimine ei kuulu automatiseerimiseks, kuna see nõuab kasutaja sisendit ja käsitsi tagasiside kogumist. Testide automatiseerimise kavandamisel on arutatud testide otstarbekkust enne nende rakendamist EbA meetodi rakenduse tooteomaniku ja kaasarendajatega.

Automatiseeritud testide koostamist oli alustatud tarkvaraga WebDriverIO. Kui alguses WebDriverIO veebilehelt juhiseid järgides tundus seadistamise protsess arusaadav ning hõlbus, siis peagi selgus, et nende testimise keskkonna üleval hoidmisel esineb probleemseid kohti. Esiteks peab WebDriverIO'l olema õige asukoht Eletctroni rakendusefailidele, kuid kui see muutub pidevalt, siis ei oska ta automaatselt neid üles leida. Teiseks ei pruugi korra jooksutatud testid enam järgmine kord õnnestuda. Seda põhjusel, et pidevalt muutuvas arendusprotsessis ning rakenduses muutuvad nupud ja nende asukohad ja seega peab neid muudatusi ka testimisel täiendavalt muutma.

Konsulteerides tooteomanikuga automatiseeritud testide teemal leiti, et II etapi töös hetkel automatiseerimise eelised ei kaalu üles nendele kuluvat aega. Automatiseeritud testid laseksid kiirelt üle kontrollida rakenduse põhifunktsionaalsusi, aga niisuguse rakenduse korral, nagu on EbA meetodi rakenduse II etapp, ei ole testitavaid elemente liigselt palju. Automatiseeritud testide kirjutamine aga nõuab palju ajalist ressurssi. Praeguses arendusprotsessis on edasi testitud rakendust käsitestimiste ning kasutatavuse testimise meetoditega, et arendusülesannetega kiirelt edasi liikuda. Rakenduse mõnel järgneval etapil on võimalik uuesti kõne alla võtta rakenduse automatiseeritud testimine ning leida efektiivne lahendus automatiseerimisele.

3.3 Rakenduse II etapi kasutatavuse testimine

Rakenduse I etapis sooritatud kasutatavuse testimisel keskenduti põhifunktsionaalsuste testimisele ja küsiti tagasisidet disainilahendusele [3, 4]. Rakenduse II etapi arenduste järel oli soov teha kasutatavuse testimine, mis sel korral hindaks lisaks arendatud funktsionaalsustele ka seda, kas rakenduse ülesehitus toetab EbA meetodiga analüüsi läbiviimist eeldatud moel. Käesolevas peatükis selgitatakse kasutatavuse testimise läbiviimist EbA meetodi rakendusele.

Kasutatavuse testimist sooritati uuringugrupis, kuhu kutsuti osalema töölauarakenduse potentsiaalseid lõppkasutajaid. Lisaks tavapärasele kasutatavuse testimisele lõppkasutajatega sooritati ka rakenduse klaviatuuril navigeerimise testimine.

3.3.1 Üldine kasutatavuse testimine

Kasutatavuse testimise läbiviimine on käesolevas lõputöös eraldatud nelja etappi:

1) planeerimine;
2) lõppkasutajate valimine;
3) kasutatavuse testimise läbiviimine;
4) testimistulemuste analüüsimine.

Kasutatavuse testimine sooritati EbA meetodi II etapi töölauarakendusega. Kasutatavuse testimisele kutsutud osalejaid nimetatakse siinses töös lõppkasutajateks või testimisel osalejateks.

Planeerimise etapis kavandati koos tooteomanikuga kasutatavuse testimise kriteeriumid, ülesanded ning vastuvõtutingimused ülesannetele. Planeerimise etapp hõlmas samuti kasutatavuse testimise teooria ja praktiliste soovitustega tutvumist ning selgeks õppimist. Selle protsessi käigus sai püstitatud uurimisküsimused, mida tahetakse teada saada kasutatavuse testimise käigus:

1. Kuivõrd intuitiivne, kasutajasõbralik on loodud II etapi töölauarakendus?
2. Kuidas mõjuvad uued lisatud muudatused, võrreldes I etapi töölauarakendusega, lõppkasutajate kasutajakogemusele?

Uurimisküsimustest lähtuvalt koostati stsenaariumipõhine ülesannete kogum ning tagasiside küsimustik. Ülesannete kogum koos ülejäänud küsimustikuga asub siinse töö Lisas 1. Lõppkasutajatele lahendamiseks antud stsenaarium kuvati neile ükshaaval. Ülesannetele järgnes lõppkasutaja iseseisev tagasisideküsimustiku täitmine. Tabelis 1 on kirjeldatud kasutatavuse testimisel esitatud stsenaariumid. Igale stsenaariumile eelneb tingimus, mis on konkreetse stsenaariumi läbimiseks vajalik. Stsenaarium loeti tehtuks, kui see vastas ka vastuvõtukriteeriumile.

Tabel 1. Kasutatavuse testimise ülesannete kriteeriumid.

Stsenaarium	**Eeltingimus**	**Vastuvõtukriteerium**
Rakenduse avamine ning uue projekti loomine.	Rakendus on alla laetud.	Rakenduses on loodud uus projekt ning nähakse ülemises vasakus nurgas, et asutakse enda loodud projektis.
Rakenduses EbA meetodiga analüüsi koostamine.	Rakenduses on loodud uus projekt ning viibitakse selle lehel.	Koostatud on vähemalt üks rida analüüsi kasutades EbA meetodit. See juures on lõppkasutaja kõiki veerge tabelis uurinud.
Rakenduses „Peida“-nimeliste elementide vajutamine.	Lõppkasutaja asub enda loodud projekti vaates.	Vajutatakse vähemalt ühte „Peida“-nimelist elementi.
Rakenduses loodud projekti allalaadimine.	Lõppkasutaja on loonud vähemalt ühe projekti rakenduses.	Lõppkasutaja laeb alla kas Exceli või JSON-formaadis projektifaili.
Rakenduses loodud projekti üleslaadimine.	Lõppkasutaja on loonud vähemalt ühe projekti rakenduses.	Lõppkasutaja laeb üles JSON-formaadis projekti ning veendub, et seda kuvatakse rakenduses.

Planeerimise etapile järgnes lõppkasutajate valimine ja kutsumine rakenduse testimisele. Lõppkasutajad kutsus testimisele töö autor ja paaril juhul tooteomanik. Testimisele kutsutud lõppkasutajad ei pidanud olema eelnevalt EbA meetodiga tuttavad, oodatud olid nii uued kui ka kogenenud kasutajad. EbA meetodi rakenduse kasutatavuse testimine eeldas, et testijad tunnevad EbA meetodi analüüsi põhimõtteid. Sellest lähtuvalt tutvustati EbA meetodit osalejatele juba testimisele kutsumise faasis ning testijatega arutati võimalikke projekte, mis sobituksid analüüsiks. Teooriaosa taustateadmistest lähtudes otsustati, et piisav arv osalejaid testimise läbiviimiseks on viis inimest, see kvootarv ka saavutati.

Enne lõppkasutajatega testimist viidi läbi kasutatavuse testimise pilootsessioon. See oli eelnev katsetus koos tooteomanikuga, et veenduda, kas kõik testimismaterjalid ja ülesanded on lõppkasutajatega testimiseks valmis. Pilootsessioon toimus videokõne teel ning tooteomanik sooritas samu ülesandeid, mis olid ette nähtud tegelikes testimissessioonides. Piloodist oli võimalik järeldada, et ülesanded olid arusaadavad, küll aga kohandati ülesannete sõnastust enne päris testimisi. Sessiooni tulemusena tuvastati ka üks programmiviga, kus rakenduse avalehel oli vaja sõnastust muuta. Antud detail edastati arendusmeeskonnale ning testimiste alguseks parandati programmiviga ära.

Testimise sessioonid toimusid kõikide osalejatega individuaalselt. Niisugune lähenemine oli vajalik, kuna nii meetodi analüüsi koostamine kui ka kasutatavuse kogemuse testimine nõuab testijalt tähelepanu ja keskendumist toimuvale. Mitme inimesega ühes ruumis ei oleks testimise tulemused enam kaardistatavad. Koos töö autoriga viibis igal testimisel samal ajal ruumis ka tooteomanik. Tooteomaniku viibimine testimisel oli vajalik, et sissejuhatavalt tutvustada osalejale EbA meetodi põhimõtteid ja abistada neid testi ajal EbA meetodi sisulise analüüsiga. Sessioonid toimusid Tartu Ülikooli Delta hoones. Enne testimisi allkirjastati osalejaga nõusoleku vorm. Testimisel paluti lõppkasutajal oma tegevusi tehes kasutada valjult mõtlemise meetodit. Testimise ajal kirjeldas lõppkasutaja oma liigutusi, nägemisi ning analüüsi tagamaid verbaalselt ning sellest koostas testimise läbiviija, käesoleva töö autor, iga sessiooni kohta päeviku-stiilis sissekande. Tabelis 2 on kirjeldatud lõppkasutajate tunnused, s.t mis on osalejate taustainfo testimise osalemise ajal. Nagu ka eelnevalt mainitud, testiti töölauarakendust viie lõppkasutajaga. Testimissessioonil kasutasid testijad oma sülearvutit, kuid kohapeal võimaldati kasutada suurt ekraani.

Tabel 2. Kasutatavuse testimisel osalenud lõppkasutajad.

Tunnus	**Lõppkasutajate arv**
Koguarv osalejaid	5
Operatsioonisüsteem	
Windows	4
macOS	1
Kogemus EbA meetodiga	
Esmakordne	5

Kasutatavuse testimise käigus kogutud märkmete põhjal koostati kokkuvõte peamistest tähelepanekutest ning arendusmeeskonnale anti suunatud soovitused edasisteks arendustöödeks. Kuna testimise põhiläbiviija ülesandeks oli uurida peamiselt EbA meetodi rakenduse kasutust, siis EbA meetodist endast tulenevaid analüüsi aspekte, mis tekitasid testijatel küsimusi, siinses lõputöös ei kajastata. EbA meetodi kasutuse tulemusi analüüsib tooteomanik ise ja teeb meetodi juhenditesse vastavad täiendused. Testimise tulemusi tutvustatakse antud töö ülejärgmises alapeatükis.

3.3.2 Kasutatavuse testimine klaviatuuriga navigeerimisel

Üldise kasutatavuse testimiste kõrval, kus testimine keskendus teatud stsenaariumipõhistele ülesannetele, sooritati ka koostöös arendusmeeskonna digiligipääsetavuse disaineriga [7]

kasutatavuse testimine rakenduse digiligipääsetavuse funktsioonile. Testimisel keskenduti klaviatuuriga liikumisele rakenduses.

Selle testimise eel allkirjastati samuti nõusoleku vorm. Testimise ülesanded koostas digiligipääsetavuse disainer ning peamisi tulemusi kajastatakse tema lõputöös [7]. Käesoleva töö autori ülesanne oli kutsuda testija kohapeale Tartu Ülikooli Delta hoonesse, jagada testimise materjalid ning anda testimise juhiseid. Testimisel jagatud materjalideks loetakse antud testimisel ülesannete kogumit, testimisel sisestatavaid andmeid ning töölauarakendust. Testimisel kasutatavad andmed koostas eelnevalt tooteomanik ning jagas neid töö autoriga. Töölauarakenduse jagamisel kasutati esialgu Linuxi versiooni, nii nagu oli testimisel osalenud lõppkasutaja isiklik arvuti. Ülesannete lahendamisel selgus, et antud Linuxi tarkvara peal ei ole kõik klaviatuuriga määratud fookuselemendid nähtavad. Fookuselementide mittekuvamise põhjust ei suudetud sessiooni ajalise piirangu tõttu seostada mõne kindla probleemiga. Sellest tulenevalt vahetati osaleja arvuti disaineri arvutiga ning jätkati testimist disaineri seadmes. Sealses keskkonnas olid klaviatuuri fookuskohad tuvastatavad ning saadi katsetada nende funktsionaalsust. Hiljem Linux Debiani rakendust testides selgus, et fookuselemente siiski kuvatakse klaviatuuriga liikumisel.

3.4 Kasutatavuse testimise tulemused

Kasutatavuse testimise tulemuste analüüsimisel eristati kaks põhikomponenti: testimise käigus kogutud märkused ja küsimustiku tulemused. Alapeatükis käsitletakse üldistavalt testimisel ilmnenud programmivigu, neist tulenevaid soovitusi ning arutelusid arendusmeeskonnaga. Lisaks kirjeldatakse küsimustiku tulemuste esitamist.

Kasutatavuse testimise sessioonide tulemusena valmis koostöös tooteomanikuga nimekiri soovitustest ja edasistest arendusettepanekutest selle kohta, mida võiks EbA meetodi töölaurakenduses täiendada. Kokkuvõte sessioonide märkmetest esitati arendusmeeskonnale tabelkujul, kus iga stsenaariumi tulemused kirjeldati üldistatuna. Stsenaariumipõhiselt on välja toodud nii positiivsed kui ka negatiivsed leiud. Vastavalt nendele koostati soovitused arendusmeeskonnale arutamiseks.

Kasutatavuse testimise tulemusena vormistati ja esitati arendustiimile ka teine, pikem loetelu parendamisvõimalustega. Need ettepanekud said läbi arutatud peale iga testimissessiooni koos tooteomanikuga ning kõigi sessioonide järel need kategoriseeriti teemade kaupa. Arendusmeeskonna arutelukoosolekul need esile kerkinud teemad prioritiseeriti. Kriitilisemad

otsustati paranda nii kiiresti kui võimalik ning kõige vähem prioriteetsemad tõsteti edasi töölaurakenduse järgmisesse etappi.

Testimissessioonidelt saadud tagasiside põhjal olid kasutajad rakendusega üldiselt rahul. Enamik stsenaariume suudeti lahendada olemasolevate visuaalsete lahendustega, mis näitab, et rakendus on suures osas intuitiivne. Ainsaks kohaks, kus lõppkasutajad vajasid täiendavat juhendamist, oli EbA meetodi analüüsi koostamine. Kuigi rippmenüüd, tekstikastid ja „info"-nupud olid kasutajatele arusaadavad, tekitas ridade lisamine segadust. See aspekt vajab kindlasti tähelepanu edasistes arendusetappides.

Windowsi versiooni allalaadimise ja käivitamisega ei esinenud kasutajatel probleeme. Küll aga tekkis takistus macOSi rakenduse puhul, kus turvahoiatus ei võimaldanud rakendust avada tavapärasel viisil. Turvaseadete käsitsi muutmine võib olla testimiskeskkonnas aktsepteeritav, kuid lõppkasutaja jaoks ei ole see kasutajasõbralik lahendus. See probleem vajab lahendamist, et tagada sujuvam ja professionaalsem kasutajakogemus.

Kasutatavuste testimise küsimustiku tulemusi tutvustati arendusmeeskonnale teemakohasel koosolekul. Küsimustiku tulemusi kuvati graafikutena ning arutati, millest võisid teatud tulemused olla tingitud. Tulemuste graafikud on jagatud ka meeskonnaliikmetega.

4. Tulemused ja arutelu

Töö viimases peatükis on praktiliste ülesannete tulemuste arutelu. Kirjeldatakse loodud lahendusi ja antakse hinnang valitud testimismeetoditele. Lisaks kirjeldatakse väljakutseid II etapi rakenduse arendamisel.

4.1 Valminud töölauarakendus

Valminud II etapi rakendus on saadaval kõigil kolmel suuremal platvormil: Windowsil, macOSil ja Linuxil. Electroni raamistikku kasutasid arenduses esmakordselt nii töö autor kui ka kompleksarendaja [6], mis tõi kaasa uusi õppimiskohti ja väljakutseid arendustöös. Järgnevad kaks alapeatükki annavad ülevaate valminud lahendusest: kust rakendust saab alla laadida, millised olid arenduse käigus esile kerkinud peamised raskused ning millised on võimalikud suunad edasiseks arenduseks.

4.1.1 II etapi töölauarakendus

Loodud II etapi töölaurakendus valmis Electroni raamistikul. Vastavalt platvormist sõltumatuse nõudele on rakendus saadaval nii Windows, macOS kui ka Linux operatsioonisüsteemil. Rakenduse funktsionaalsused on säilinud igal platvormil. Kõik versioonid rakendusest on alla laetavad II etapi arenduse repositooriumist Bitbucket[23].

Alla laetava rakenduse failitüüp sõltub operatsioonisüsteemist: Windowsi jaoks on failil laiend „.exe“, macOSi jaoks „.dmg“ ning Debian Linuxi jaoks „.deb“. Windowsi versiooni lõi kompleksarendaja [6]. MacOSi ja Linuxi failide koostamine oli siinse töö autori ülesanne. MacOSi puhul on arvestatud ka riistvaralisi erinevusi: eraldi failid on loodud nii Apple'i M-seeria kiipidega arvutitele kui ka Intel-protsessoriga seadmetele. Debian Linuxi puhul on rakendused eristatud arhitektuuri alusel – „arm64“ ja „x64“.

Faili allalaadimisel on oluline, et lõppkasutaja valiks oma seadmele sobiva versiooni, et tagada rakenduse korrektne toimimine. MacOSi ja Linuxi versioonid võtavad arvutis ligikaudu 700 MB kettaruumi. Rakenduse esmakordsel käivitamisel luuakse kasutaja „Dokumentide“ kausta alamkaust nimega EbA-UIM. Sinna salvestatakse rakenduse andmebaasifail ning logifail, mida kasutatakse töölauarakenduse tööks ja veaotsinguks. MacOSis võib rakenduse avamisel ilmuda turvahoiatus, kuna rakendus ei ole varustatud Apple'i ametliku digitaalallkirjaga. Sellisel juhul ei luba süsteem rakendust kohe tavapärasel viisil avada. Rakenduse esmakordseks

[23] EbA-UIM II etapi rakenduse allalaadimise link: https://bitbucket.org/eba-method/eba-code/downloads/

käivitamiseks tuleb kasutajal käsitsi turvaseadetest lubada selle avamine. Pärast esmast lubamist saab rakendust edaspidi avada tavapärasel viisil.

4.1.2 Väljakutsed

Rakenduse II etapi üheks olulisemaks väljakutseks siinse töö autorile osutus tagasüsteemi käivitamine macOS operatsioonisüsteemil. Probleem tulenes asjaolust, et macOSis on rangemad turvareeglid – nn aediku põhimõtted – võrreldes näiteks Windowsiga. Kuna tagasüsteem vajab töötamiseks võimalust luua andmebaasifail, siis ei lubanud macOS seda teha rakenduse käivitamise kaustas. Probleemi lahendamiseks suunati tagasüsteem kasutama failide salvestamiseks kasutaja „Dokumendid“ kausta, mis vastab süsteemi turvanõuetele.

Tulenevalt andmebaasifaili uude asukohta suunamisest ei ole II etapi rakenduses enam võimalik kohe pärast avamist kuvada EbA meetodi näidisprojekti, nagu see oli I etapis. Näidisprojekt on aga oluline selleks, et kasutaja saaks kiire ülevaate rakenduse toimimisest. Kuna rakenduse esmakordsel käivitamisel luuakse täiesti uus andmebaasifail, ei sisalda see vaikimisi näidisandmeid. Küll aga on kasutajal võimalus näidisprojekt ise rakendusse laadida JSON-formaadis. Vastav fail on saadaval nii eesti- kui ingliskeelsena rakenduse allalaadimislingi[23] juures.

Rakenduse II etapi kasutatavuse testimise käigus ilmnes probleem macOS platvormil, kus rakenduse esmakordsel käivitamisel kuvatakse turvahoiatus. See on tingitud sellest, et töölauarakendusel puudub Apple’i digitaalne allkiri. Hoiatusest möödumiseks peab kasutaja rakenduse turvaseaded käsitsi muutma, mis ei ole kasutajakogemuse seisukohalt kuigi mugav. Kasutajasõbralikumaks lahenduseks oleks rakendusele digiallkirja lisamine juba pakendamise etapis, mis võimaldaks vältida täiendavaid samme rakenduse käivitamisel.

4.2 Hinnang rakenduse testimisteks valitud meetoditele

Rakenduse II etapi arenduses sai rakendatud kahte metoodikat, nii käsitestimist kui ka kasutatavuse testimist. Käsitestimise sooritas töö autor rakenduse II etapi arendustsüklite ajal, vastavalt testitavatele muudatustele. Rakenduse kasutatavuse testimiseks viidi läbi üldine testimine viie lõppkasutajaga ning klaviatuuriga navigeerimise testimine ühe kasutajaga.

4.2.1 Hinnang käsitestimise meetodile

II etapi rakenduse testimiseks kasutatud suvatestimise ja regressioontestimise hübriidlahendus sobitus hästi agiilsesse arendustsüklisse. Arendajate tehtud muudatused testiti iganädalaselt

enne meeskonnakoosolekuid. Vajadusel anti arendajatele tagasisidet Jira kaudu või muude sobivate töövahendite abil.

Testimine keskendus peamiselt rakenduse visuaalsele poolele. Kuigi tutvuti ka muudatuste koodiosadega, ei mõjutanud see kordagi testitulemusi otseselt. Peamine eesmärk oli hinnata, kas muudatused vastasid kokkulepitud vastuvõtukriteeriumitele ning sobitusid rakenduse üldise loogika ja välimusega.

4.2.2 Hinnang kasutatavuse testimise meetodile

Kasutatavuse testimisel lähtuti töö teoreetilises osas esitatud soovitustest. Lõppkasutajatel paluti sooritada testülesandeid, rakendades seejuures valjult mõtlemise meetodit. Ülesanded simuleerisid realistlikke olukordi, millega rakenduse sihtrühm tõenäoliselt kokku puutub. Pärast testimist tutvustati tulemusi arendusmeeskonnale.

Osalejate valikul peeti oluliseks, et nad esindaksid rakenduse võimalikke tegelikke kasutajaid. See aitas tagada testimise käigus saadud tagasiside asjakohasuse ja kvaliteedi. Teooriaosas soovitatud viie osaleja miinimumnõue sai täidetud. Selline osalejate arv oli sobilik, kuna iga testija tõi esile uusi tähelepanekuid, ent juba varasemalt kaardistatud probleemid ei hakanud liialt korduma.

Testimisülesannete koostamisel peeti oluliseks, et need oleksid realistlikud ja kasutajale huvipakkuvad, mis soodustas osalejate kaasatust ja aitas vältida väsimust. Kõik testimised viidi läbi kohapeal, kus läbiviijad ja osaleja viibisid samas ruumis. See oli käesoleva töö raames otstarbekas, kuna rakenduse käivitamisel võis esineda tehnilisi tõrkeid ning samas sai osalejatele tutvustada EbA meetodi olemust. Valjult mõtlemise meetod osutus sobivaks, kuna see võimaldas töö autoril ja tooteomanikul koguda sisulist tagasisidet rakenduse kasutusmugavuse ja funktsionaalsuse kohta.

Kuna testimise eesmärk oli eelkõige kvalitatiivne, esitati tulemused arendusmeeskonnale kirjeldades kohapeal tekkinud olukordi. See võimaldas meeskonnal saada väärtuslikku sisendit edasiseks arendustööks. Küsimustikust kogutud andmed olid pigem kvantitatiivse iseloomuga ning nende tulemusi esitleti graafiliselt, kooskõlas teooriaosas käsitletud põhimõtetega.

Kokkuvõte

Bakalaureusetöö eesmärgiks oli arendada edasi EbA meetodi I etapi töölauarakendust ning läbi viia erinevaid testimisi. Loodud II etapi rakendus valmis meeskonnatööna ning on kättesaadav kolmel operatsioonisüsteemil – Windows, macOS ja Linux. Rakenduse testimiseks sooritati hulk erinevaid käsitestimisi ning lõppkasutajatega kasutatavuse testimine.

Edasiarendatud EbA-UIM rakendus valmis Electroni raamistiku põhjal, mis võimaldas luua platvormist sõltumatu töölauarakenduse, säilitades samal ajal suure osa I etapis välja töötatud struktuurist. Electroni raamistikule üleviimisel kõik eelnevalt arendatud põhifunktsionaalsused säilisid ning II etapi muudatused kohandati kõigile toetatud platvormidele. Töö autori ülesandeks oli tagada rakenduse toimimine nii macOSi kui ka Linuxi operatsioonisüsteemides.

Arenduse käigus ilmnes, et macOSi rakendustele kehtivad oluliselt rangemad turvanõuded kui teistel platvormidel. Kuna Electroni kasutamine oli uus kogemus, kujunes see esialgu väljakutseks. Siiski leiti sobiv lahendus, mis võimaldas rakenduse turvanõuetele vastavaks muuta ja seejärel kasutajatele macOSil kättesaadavaks teha.

Töö autor viis läbi käsitestimisi, et kontrollida uusi muudatusi vastavalt eelnevalt kokkulepitud vastuvõtukriteeriumitele. Teised arendusmeeskonna liikmed suunasid tehtud arendused töö autorile testimiseks ja tulemuste põhjal said arendajad kiire tagasiside, mis toetas tõhusat ning agiilset arendusprotsessi.

II etapi arendustööde lõpus läbiviidud kasutatavuse testimine andis väärtuslikku tagasisidet olemasolevate funktsionaalsuste kohta. Testimisel rakendatud valjult mõtlemise meetod osutus tõhusaks lõppkasutajate tegevuste analüüsimisel, võimaldades paremini mõista nende mõttemustreid ja kogemusi. Testimise tulemuste põhjal koostati soovitused rakenduse edasiseks arendamiseks.

EbA-UIM II etapi rakendus on alla laetav programmikoodi Bitbucketi repositooriumist[23].

Viidatud kirjandus

[1] Piirisild A., Perandrés Gómez A., Taveter K. A New Usability Inspection Method: Experience-Based Analysis. Requirements Engineering: Foundation for Software Quality, 2024. https://doi.org/10.1007/978-3-031-57327-9_5 (8.12.2024).

[2] Riigikohus. Siim Tammeri kaebus Politsei- ja Piirivalveameti 1. veebruari 2023. a otsuse peale, 3-23-462, 2025, 20.6. (29.04.2025)

[3] Kuldmaa K. O. Rakendusprogrammi loomine „Kasutatavuse kogemuspõhise analüüsi meetodile“. TÜ arvutiteaduste instituudi bakalaureusetöö. 2024. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=79625 (8.12.2024).

[4] Kreinin I. Kasutajaliidese disaini loomine „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele. TÜ arvutiteaduste instituudi bakalaureusetöö. 2024. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=79737 (8.12.2024).

[5] Kreinin I., Kuldmaa K. EUIM. GitHub repositoorium. 2024. https://github.com/karl1245/EUIM (8.12.2024)

[6] Põldma, S. „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakenduse edasiarendus ja tiimitöö juhtimine. TÜ arvutiteaduste instituudi bakalaureusetöö. 2025. (11.05.2025)

[7] Pettai, R. Kasutajaliidese disaini ja digiligipääsetavuse edasiarendus „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele. TÜ arvutiteaduste instituudi bakalaureusetöö. 2025. (11.05.2025)

[8] Müller E. Web Technologies on the Desktop: An Early Look at Flutter. University of Stuttgart bakalaureusetöö. 2021. https://elib.uni-stuttgart.de/bitstream/11682/11515/1/thesis.pdf (8.12.2024).

[9] Alymkulov D. DESKTOP APPLICATION DEVELOPMENT USING ELECTRON FRAMEWORK Native vs. Cross-Platform. South-Eastern Finland University of Applied Sciences bakalaureusetöö. 2019. https://www.theseus.fi/bitstream/handle/10024/167669/Daniyar_Alymkulov.pdf (8.12.2024).

[10] Itkonen J., Mantyla M. V., Lassenius C. "How do testers do it? An exploratory study on manual testing practices," 2009 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, USA, 2009, pp. 494-497, doi: 10.1109/ESEM.2009.5314240. (29.04.2025)

[11] Agruss, C., Johnson, B. Ad Hoc Software Testing A perspective on exploration and improvisation. 2000. https://www.onestoptesting.com/testing-download/download/Ad%20Hoc%20Testing.pdf (29.04.2025).

[12] Bhatti I., Siddiqi J.A., Moiz A., Memon Z.A. Towards Ad hoc Testing Technique Effectiveness in Software Testing Life Cycle. 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). 2019 Jan;1–6. (29.04.2025)

[13] Garousi V., Mäntylä M. V. When and what to automate in software testing? A multi-vocal literature review. 2016. https://www.sciencedirect.com/science/article/pii/S0950584916300702#sec0001 (8.01.2025)

[14] Pelivani E., Cico B., "A comparative study of automation testing tools for web applications," 2021 10th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2021, pp. 1-6, doi: 10.1109/MECO52532.2021.9460242. (29.04.2025)

[15] Rose E. J. Usability Testing Plan Template: A flexible tool for planning and teaching usability evaluation. 2022. https://dl.acm.org/doi/pdf/10.1145/3548658 (8.12.2024).

[16] Moran K. Usability (User) Testing 101. 2019. https://www.nngroup.com/articles/usability-testing-101/ (8.12.2024).

[17] Budiu, R. Quantitative vs. Qualitative Usability Testing. 2017. https://www.nngroup.com/articles/quant-vs-qual/ (15.05.2025)

[18] Nielsen J. How Many Test Users in a Usability Study? 2012. https://www.nngroup.com/articles/how-many-test-users/ (29.04.2025)

[19] Nielsen J. Thinking Aloud: The #1 Usability Tool. 2012 https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/ (29.04.2025)

[20] Kõiv E. Valjult mõtlemise meetod: vahend eneseregulatsioonioskuse toetamiseks kognitiivsete ja metakognitiivsete õpistrateegiate kaudu. TÜ Haridusteaduste instituudi magistritöö. 2019. https://dspace.ut.ee/server/api/core/bitstreams/bbfa3082-dadd-42d4-896e-34c0ca733837/content (29.04.2025)

[21] Lewis J. R. Handbook of Human Factors and Ergonomics, Fourth Edition Chapter 46: Usability Testing. 2012. https://doi.org/10.1002/9781118131350.ch46 (29.04.2025)

[22] Rubin, J., Chisnell, D. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. Germany: Wiley, 2011

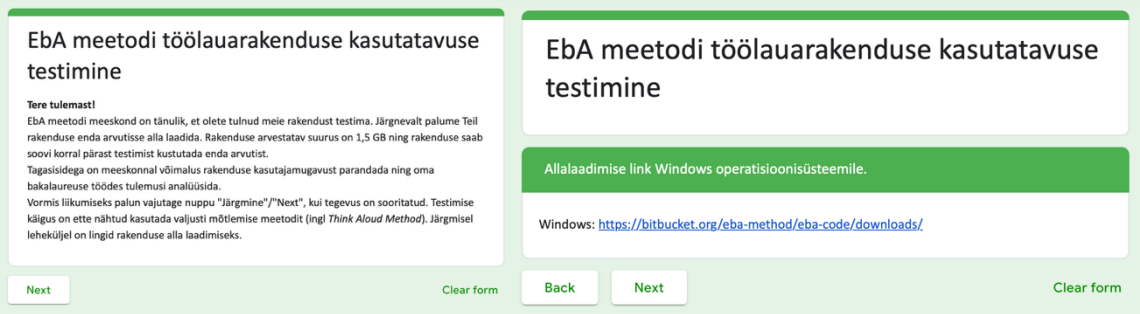
[23] Andmekaitse Inspektsioon. 9. peatükk. Nõusoleku küsimine. 2024. https://www.aki.ee/9-peatukk-nousoleku-kusimine (15.05.2025)

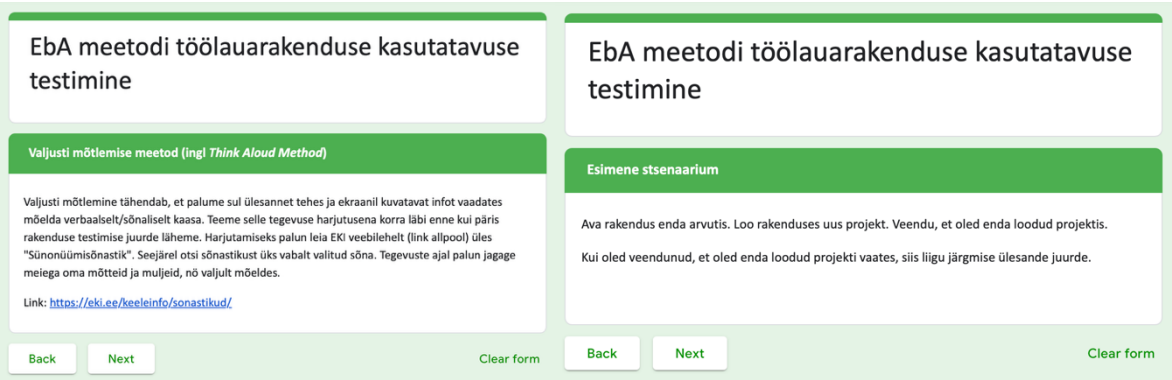
[24] Nielsen J., Usability Engineering. Cambridge, MA: Academic Press, Inc., 1993.

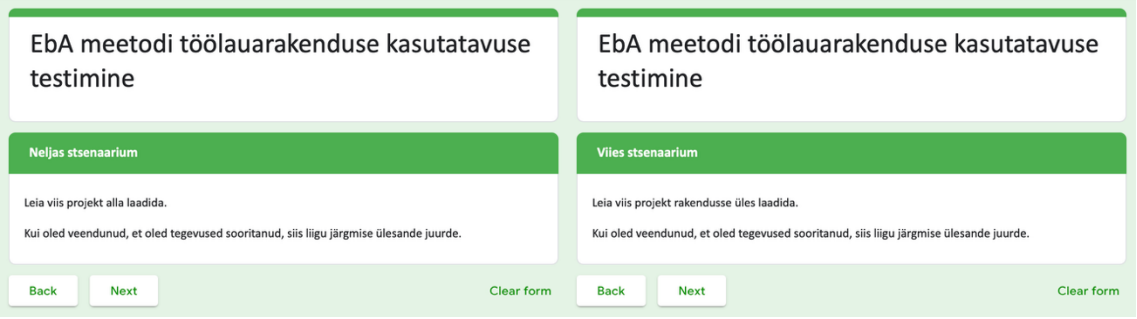
Lisad

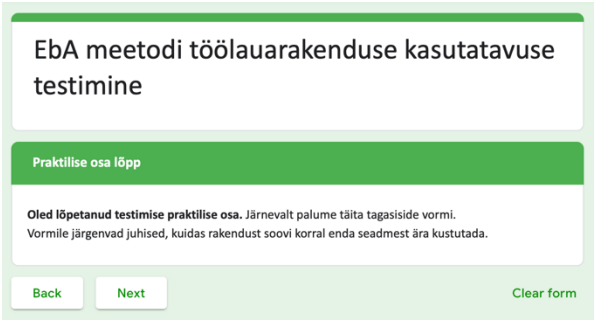
I. Kasutatavuse testimise küsimustik.

Lisas 1 on kasutatavuse testimise küsimustik, mis koosneb ülesannetest ja tagasiside küsimustikust. Küsimustiku koostas töö autor koostöös tooteomanikuga. Küsimustiku täismahus andmed on jagatud rakenduse II etapi arendusmeeskonnaga.





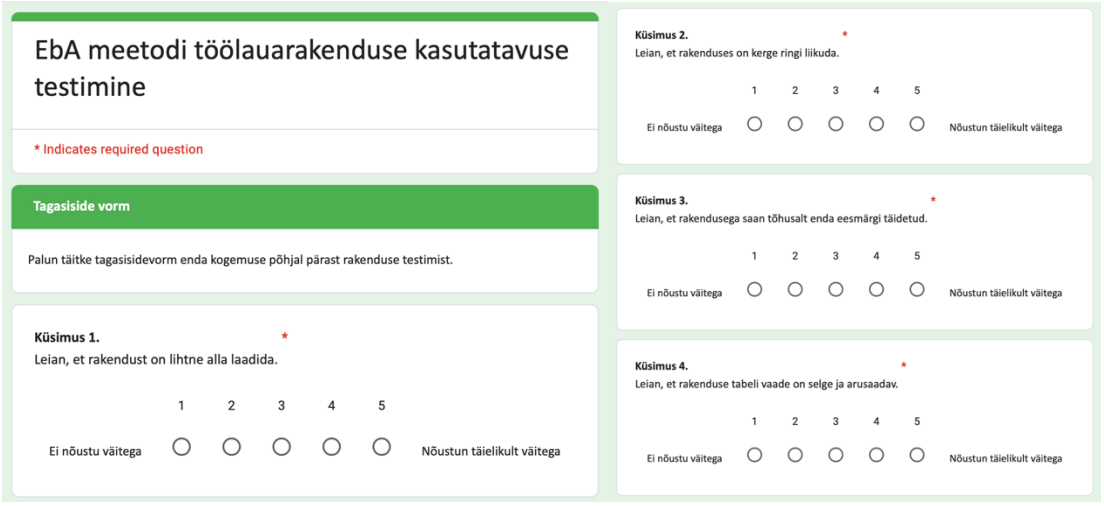



EbA meetodi töölauarakenduse kasutatavuse testimine

Praktilise osa lõpp

Oled lõpetanud testimise praktilise osa. Järnevalt palume täita tagasiside vormi.
Vormile järgenvad juhised, kuidas rakendust soovi korral enda seadmest ära kustutada.

Back Next Clear form


EbA meetodi töölauarakenduse kasutatavuse testimine

* Indicates required question

Tagasiside vorm

Palun täitke tagasisidevorm enda kogemuse põhjal pärast rakenduse testimist.

Küsimus 1. *
Leian, et rakendust on lihtne alla laadida.

	1	2	3	4	5	
Ei nõustu väitega	○	○	○	○	○	Nõustun täielikult väitega

Küsimus 2. *
Leian, et rakenduses on kerge ringi liikuda.

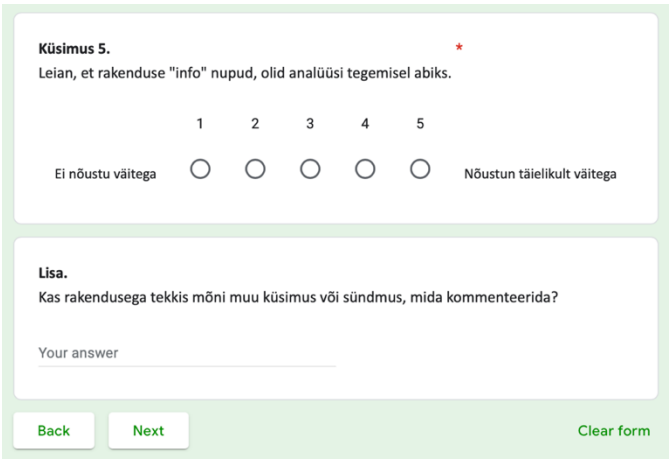
	1	2	3	4	5	
Ei nõustu väitega	○	○	○	○	○	Nõustun täielikult väitega

Küsimus 3. *
Leian, et rakendusega saan tõhusalt enda eesmärgi täidetud.

	1	2	3	4	5	
Ei nõustu väitega	○	○	○	○	○	Nõustun täielikult väitega

Küsimus 4. *
Leian, et rakenduse tabeli vaade on selge ja arusaadav.

	1	2	3	4	5	
Ei nõustu väitega	○	○	○	○	○	Nõustun täielikult väitega

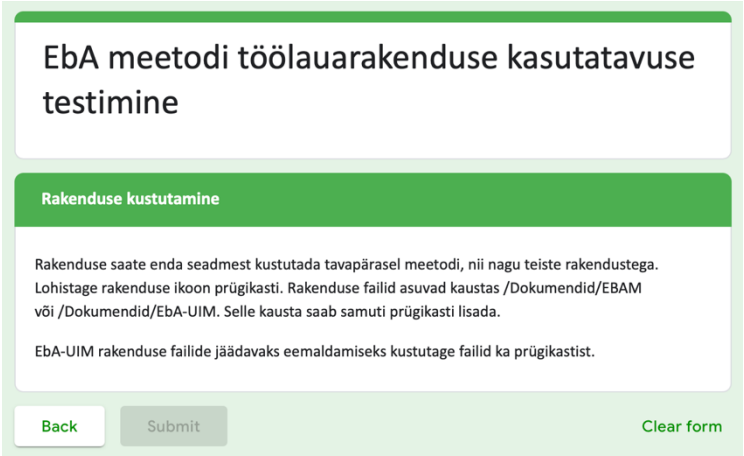

Küsimus 5. *
Leian, et rakenduse "info" nupud, olid analüüsi tegemisel abiks.

	1	2	3	4	5	
Ei nõustu väitega	○	○	○	○	○	Nõustun täielikult väitega

Lisa.
Kas rakendusega tekkis mõni muu küsimus või sündmus, mida kommenteerida?

Your answer

Back Next Clear form


EbA meetodi töölauarakenduse kasutatavuse testimine

Rakenduse kustutamine

Rakenduse saate enda seadmest kustutada tavapärasel meetodi, nii nagu teiste rakendustega. Lohistage rakenduse ikoon prügikasti. Rakenduse failid asuvad kaustas /Dokumendid/EBAM või /Dokumendid/EbA-UIM. Selle kausta saab samuti prügikasti lisada.

EbA-UIM rakenduse failide jäädavaks eemaldamiseks kustutage failid ka prügikastist.

Back Submit Clear form

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Gerdo Germann,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Testimised ja macOSile kohandamine „Kasutatavuse kogemuspõhise analüüsi meetodi“ rakendusele**, mille juhendaja on Anu Piirisild, 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.

Gerdo Germann

15.05.2025