TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Kadri Jõgi 193599IADB

Vaimse tervise toetamise veebirakenduse IntroScope prototüüp

Bakalaureusetöö

Juhendaja: Kristiina Hakk

PhD

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen käesoleva bakalaureusetöö ise koostanud ja seda pole keegi teine varem kaitsmisele esitanud. Kõik lõputöös kasutatud muudest allikatest pärit andmed, olulised seisukohad ja teiste autorite tööd on viidatud.

Autor: Kadri Jõgi

16.05.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua vaimset tervist toetava veebirakenduse IntroScope prototüüp. Kavandatava rakenduse mõte on pakkuda rakenduse tulevastele kasutajatele võimalust end iseseisvalt analüüsida ning suurendada seeläbi toimetulekut igapäevase vaimse pingega. Samuti on võimalik rakendust kasutada koostöös psühholoogiga teraapia kodutööde jagamiseks ja haldamiseks.

Töö käigus arendatakse veebirakenduse prototüüp, mis võimaldab kasutajatel isikliku konto kaudu ligipääsu eri eneseanalüüsi vormidele ja varasematele sisestustele ning nõustaja rollis olevatel kasutajatel klientide kodutöödele.

Lõputöö rõhuasetus on kasutajakogemuse ja -liidese disainil, töö käigus analüüsitakse selle valdkonna põhilisi vigu ning parimaid praktikaid ning püütakse rakenduse arenduse käigus neist lähtuda.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 46 leheküljel, 7 peatükki, 22 joonist, 4 tabelit.

Abstract

A Prototype for Mental Health Support Web Application IntroScope

The purpose of the current thesis is to create a prototype for a mental health support web application IntroScope. The goal of this application is to provide support for potential users to monitor themselves and thus be better able to cope with the daily stress. It can also be used in a psychological therapy process to manage and share homework.

In completion of the thesis, a prototype is going to be developed that enables users through personal accounts to access their previous evaluations and start new ones. Counselors can also access their client´s homework that they have shared with them.

The emphasis of this thesis is on user experience and user interface design. The best practices and main mistakes of these fields are going to be considered and the gained insights are going to be applied in the development process.

The thesis is in Estonian and contains 46 pages of text, 7 chapters, 22 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* – rakendusliides.
BLL	*Business Logic Layer* – rakenduse äriloogika kiht.
CSS	*Cascading Style Sheets* – veebilehtede küljendamisel kasutatav märgistuskeel.
DAL	*Data Access Layer* – andmevahetuskiht.
DRY	*Don´t Repeat Yourself,* programmeerimise põhimõte, mille kohaselt tuleks vältida tarbetuid koodikordusi.
DTO	*Data transfer object*, andmeedastusobjekt programmeerimises.
Eepos	ingl k *epic* – mahukas kasutajalugu.
ERD	*Entity Relationship Diagram* – andmebaasi disaini struktuuri kujutav diagramm.
HTTP	*Hypertext Transfer Protocol* – interneti protokoll hajusate infosüsteemide vaheliseks andmevahetuseks.
IDE	*Integrated Development Environment* – interaktiivne arenduskeskkond, koodiredaktor.
JSON	*JavaScript Object Notation* – andmemahu mõttes kerge andmevahetusformaat, mis on ühteaegu nii inim- kui ka masinloetav.
MVC	*Model-View-Controller* – mudel-vaade kontroller, rakenduse arhitektuurimuster.
MVP	*Minimal Viable Product* – rakenduse minimaalsete funktsioonidega toimiv versioon.
ORM	*Object Relational Mapping* – objekt-relatsioonvastendamine, andmeobjektide ühest teiseks muutmine.

POCO	*Plain Old C# Object* – lihtsad andmeobjektid, mida kasutatakse suhtluses andmebaasiga ning mille raames ei toimu äriloogilisi tegevusi.
POST päring	HTTP päringu üks meetodeid, mida veebitehnoloogiad andmevahetuseks kasutavad. POST päring eeldab, et veebiserver võtab vastu päringu sõnumis sisalduvad andmed.
SDK	*Software Development Kit*, tarkvara arenduse komplekt.
SQL	*Structured Query Language*, andmebaasi päringukeel.
UI	*User Interface* – kasutajaliides.
UML	*Unified Modeling Language* – ühtne või standardne modelleerimiskeel.
UX	*User Experience* – kasutajakogemus.
WCAG	*Web Content Accessibility Guidelines* – veebisisu juurdepääsetavuse juhised. Osa veebi juurdepääsetavuse juhistest, mille avaldas Interneti-põhise rahvusvahelise standardiorganisatsiooni World Wide Web Consortium Web Accessibility Initiative.
Web API	Võrgupõhine rakendusprogrammiliides.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

COVID-19 pandeemia ja sellega seotud elustiilimuutused on toonud endaga kaasa vaimse tervise probleemide tõusutrendi [1]. Üha enam inimesi teadvustab, et on kogenud läbipõlemist, ärevust, depressiooni, keskendumisraskusi või muid vaimse tervise probleeme [2]. Riiklikult rahastatud psühholoogi vastuvõtule saamine võtab pikkade järjekordade tõttu kaua aega ning erapraksistes pakutav tasuline teraapia pole kõikidele majanduslikult jõukohane [3]. See tekitab olukorra, kus abivajajad ei pruugi õigel hetkel abi saada.

Terapeudid kasutavad teraapia käigus sageli kodutöid, mida klient seansiväliselt iseseisvalt eneseanalüüsiks teeb [4]. Osad kergemad vaimse tervise probleemid oleksid ennetatavad või leevendatavad, kui klientidel oleks võimalik enda vaimset tervist iseseisvalt paremini analüüsida [5]. Uurimused on näidanud, et näiteks teadvelolek võib aidata märkimisväärselt ärevuse ja depressiooni sümptomeid vähendada ning korduva depressiooni korral terve püsida ning tagasilööke vältida [6]. Hetkel kasutatakse teraapia käigus eneseanalüüsiks tihtipeale paberkandjal eneseanalüüsi vorme ja küsimustikke [7][8], kuid puudub töövahend, kuhu saaks pidevalt märkmeid tehes protsessist parema ülevaate.

Käesoleva töö eesmärk on uurida vaimse tervise spetsialistidelt, milliseid kodutöid nad klientidele soovitavad ning luua rakendus, kuhu need küsimustikud ja vormid koondada, et soodustada iseseisvat tööd vaimse tervise toetamiseks. Loodav veebirakendus võimaldab ligipääsu eneseanalüüsi vormidele, mida kasutaja saab soovi korral iseseisvalt või teraapia käigus vaimse tervise toetamiseks täita.

Töö teoreetilises osas käsitletakse psühholoogilisi uuringuid eneseanalüüsi mõjust vaimse tervise edendamisele. Samuti lahatakse olemasolevaid lahendusi ja võimalusi, tuuakse välja eri analüüsivormide ühis- ja eriosa ning analüüsitakse planeeritava rakenduse tehnoloogiate valikut, sh raamistikku ja programmeerimiskeelt, töökeskkonda, andmebaasisüsteemi ja versioonihaldust.

Seejärel analüüsitakse tulevase rakenduse funktsionaalseid ja mittefunktsionaalseid nõudeid, koostatakse kasutusmallide mudel ning määratakse rakenduse skoop. Töö tulemusel valmib vaimset tervist toetava veebirakenduse prototüüp.

2 Ülevaade probleemist

Vaimse tervise probleemid on olnud pikka aega stigmatiseeritud staatuses valdkond, kus on nähtud suurt vaeva, et suurendada ühiskondlikku teadlikkust ja võtta kasutusele ennetavaid meetmeid [9]. Probleemi on võimendanud COVID-19 pandeemia, mis tõi igapäevaellu uued stressiallikad ning kõrgendatud läbipõlemisriski paljudel elualadel, eelkõige eesliinitöötajatel [10]. Lisaks vähenes riiklike kontaktipiirangute tõttu abi kättesaadavus ning suurenes üldine isiklik isoleeritus. Professionaalse abi saamist raskendavad pikad järjekorrad riiklikele teenustele ja erapraksiste küllaltki kõrged hinnad [11].

Terapeudid, kelle vastuvõtule üldjuhul esmalt pöördutakse, kasutavad töös mitmeid eneseanalüüsi vorme ja küsimustikke [12]. Neid võidakse soovitada teraapiavälisel ajal kodus täitmiseks. Tihtipeale on need paberkandjal, mistõttu on raske andmeid hallata ja neist kokkuvõtteid teha. Lisaks kaasnevad paberkandjal andmete hoidmisega mitmed andmeturbe probleemid [13]. Näiteks käideldavuskadu, kui oluline ankeet kuhugi ära kaob. Samuti on probleemiks konfidentsiaalsus, kui näiteks tundliku infoga märkmeid hoitakse ühiskasutatavate ruumide kappides, mida tuleb tihtipeale ette näiteks koolipsühholoogidel, kes teenindavad üldjuhul mitmeid koole korraga ning neil ei pruugi olla eraldi isiklikku kabinetti. Täidetud paberkandjal töölehe puhul on ka see puudus, et andmeid on vaid üks eksemplar, aga kui nii terapeut kui ka klient sooviksid enda arengut ülestähenduste abil jäädvustada, siis tuleks hakata neid lehti kopeerima, mis on üsna tülikas. Terapeudi jaoks oleks hea monitoorida kliendi arengut ja täidetud töölehte enne seanssi, et seansi aega tõhusamalt kasutada.

2.1 Psühholoogiline säilenõtkus

Igaüks kogeb elu jooksul vaimset pinget, ärevust, masendust, sihitust, mis võivad esineda ka siis, kui tõsisema psüühikahäire diagnoosimiseks ei ole alust [14]. Tervise Arengu Instituudi väitel on tavaliselt sel puhul tegu negatiivse stressi tagajärgedega, mis võivad

aja jooksul kuhjuda ja süveneda, olles nii riskiteguriks psüühikahäire kujunemisele. Riskiteguriteks võivad olla näiteks pikaajalised halvad toimetulekumehhanismid, nagu halb unehügieen, ebatervislik toitumine, vähene liikumine, liigne stress, sõltuvust tekitavate ainete kuritarvitamine, negatiivsed mõttemustrid ja liigne nutiseadmete kasutamine [15]. Kui näiteks ühe magamata öö tõttu ei pruugi midagi korrast ära minna, siis pideva ebapiisava une korral on uuringud leidnud märkimisväärse seose suurema negatiivsena tunnetatud vaimse pingega [16].

Samuti võib igaüht tabada ootamatu kriis, nagu töö või lähedase kaotus, õnnetus või tõsine haigus. Neid eelmainitud muutusi ja stressoreid kogetakse erinevalt, kuid enamike elumuutvate olukordadega siiski kohanetakse. Psühholoogid nimetavad võimet stressirohkete olukordadega kohaneda säilenõtkuseks (*resilience*). Kuigi raskused võivad olla ootamatud ja valusad, ei pea need tingimata elu „rööpast välja lööma“. Alati leidub elus tahke, mida saab kontrollida, muuta või arendada. Siinkohal on otsustav roll säilenõtkusel. See mitte üksnes ei aita keerulistest olukordadest välja tulla, vaid ka elukvaliteeti tõsta. [17]

Kuigi osad inimesed on teistest säilenõtkemad, ei tähenda see, et säilenõtkus oleks fikseeritud isikuomadus. On leitud, et see kätkeb eri käitumisviise, mõtteid ja tegusid, mida on võimalik ajapikku ja eesmärgipäraselt arendada. American Psychological Associationi väitel on selle jaoks olulised neli põhilist mõjutegurit [17]:

1. **Suhtevõrgustiku loomine** – see hõlmab suhete prioriseerimist, grupikuuluvust, abi vastuvõtmist, enda ümbritsemist inimestega, kes on kaastundlikud ja aktsepteerivad.
2. **Heaolu arendamine** – muuhulgas füüsiline tervis, näiteks tervislik toitumine, piisav uni ja liikumine. Lisaks teadveloleku harjutamine, nagu päeviku pidamine, jooga, meditatsioon või muud vaimsed praktikad. Samuti negatiivsete toimetulekumehhanismide vältimine (näiteks sõltuvust tekitavate ainete abil stressi leevendamisest hoidumine).
3. **Eesmärkide seadmine** – näiteks regulaarne oskuste arendamine, ettevõtlikkus mõne probleemi lahendamisel, sihipärane eesmärkide poole liikumine.
4. **Eluterve mõtteviisi kujundamine** – nagu mõtlemisvigade tuvastamine, positiivse sisekõne juurutamine, muutuste aktsepteerimine ning optimistliku ellusuhtumise harjutamine.

Igasugune harjumuste kujundamine on õppimismehhanism [18]. Psühholoog Wendy Wood, kes uuris harjumuste kujunemist, leidis, et kui midagi positiivset piisavalt tihti edukalt korrata, muutub selle tegemine ajapikku lihtsamaks ja automaatsemaks ning närvirakkude tasandil tekib n-ö otsetee [18]. Harjumuste kujunemisel on olulisel kohal enda käitumise analüüs. Edu saavutamiseks tuleb osata hinnata hetkeolukorda ning seada realistlikke eesmärke [19].

2.2 Lõputöö eesmärk

Käesoleva lõputöö eesmärk on luua prototüüp veebirakendusest, mis oleks abivahend positiivse vaimse tervise edendamiseks koostöös psühholoogiga teraapiavälisel ajal või iseseisvaks kasutamiseks, et suurendada säilenõtkust, kujundades tervislikke toimetulekuharjumusi ja leevendades negatiivset stressi ning sellega kaasneda võivate vaimse tervise probleemide ilmnemist.

Rakendus on mõeldud olema toeks heade harjumuste kujundamisel, et motiveerida seatud eesmärke saavutama, kuni positiivsed käitumismustrid jõuavad automatiseeruda ning nende regulaarne kordamine ei nõua enam nii palju pingutust ning oht tagasilanguseks väheneb. Samuti on see mõeldud hõlbustama nõustaja tööd kliendi kodutööde haldamisel, muutes teraapia läbiviimise tõhusamaks, kuna see võimaldab kodutöödest ülevaade saada juba enne teraapiaseanssi.

Prototüübi all mõeldakse antud juhul toimiva funktsionaalsusega veebirakenduse MVP-d (*Minimal Viable Product*) ehk rakendus täidab lõputöö valmides kõiki funktsioone ühe väljatöötatud eneseanalüüsi vormi näitel ning sinna on võimalik edasiarenduste käigus eneseanalüüsi vorme juurde luua ning kogu rakendus avalikult kättesaadavaks teha.

3 Olemasolevad lahendused

Vaimset tervist toetavaid abivahendeid on juba praegu mitmeid. Leidub nii eesti- kui inglisekeelseid materjale, paberkandjal ja digitaalseid lahendusi. Järgmisena analüüsitakse mõnede saadaolevate võimaluste eeliseid ja puudusi.

3.1 Peaasi.ee

Üheks virtuaalseks teabeallikaks on näiteks riiklikult rahastatud noortele suunatud vaimse tervise portaal peaasi.ee, kus on võimalik leida mitmeid juhendmaterjale teksti ja videote kujul. Portaal on kaasaegse kujundusega ning kasutab eriala spetsialistide soovitustel põhinevaid lähenemisviise. Seal on võimalik saada kohest anonüümset nõustamist, kustkaudu suunatakse vajadusel hiljem ka teraapiasse. Portaali kaudu reklaamitakse mitmeid koolitusi nii eraisikutele kui ka ettevõtetele, sh tasulisi koolitusi. Abi leiab vaimse tervise edendamisel, aga ka kohest sekkumist vajava kriisi lahendamisel (nt Eluliin, ohvriabi, kiirabi). Portaalis saab alla laadida ka vaimse tervise edendamise töövihikuid (“Minu vaimse tervise esmaabi” ja “Ärevuse töövihik”), kuid puudub võimalus sisseloginuna neid virtuaalselt täita. Portaali kaudu on võimalik neid materjale endale trükistena tasuta koju tellida, kuid paberkandjal töövihikus on võimalik täita lahtreid vaid üks kord, mis välistab võimaluse monitoorida arengut pikema aja jooksul.

3.2 Terviseinfo.ee

Portaal keskendub rohkem teaduspõhisele teadlikkuse suurendamisele eri tervisega seotud teemadel (toitumine, alkohol, seksuaaltervis, vähk jne), sh vaimne tervis. Vaimse tervise valdkonnas on välja antud palju trükiseid eesti ja ka vene keeles. Näiteks stressi, enesekehtestamise, mõtlemise, enesehinnangu kohta. Adresseeritakse eri vanusegruppe – koolieelikuid, noori, eakaid. Kõik materjalid on PDF-formaadis, neid saab tasuta alla laadida, kuid ühtegi interaktiivset lahendust nende täitmiseks ei pakuta.

3.3 Vaimnetervis.ee

Vaimnetervis.ee on Tallinna Vaimse Tervise Keskuse portaal. Ka seal leidub spetsialistide nõuandeid, kuidas toetada vaimset tervist. Sealtkaudu antakse infot võimalike (nii tasuliste kui ka tasuta) teenuste kohta. Portaali sisu on pigem kesine, rõhk asutuse ja selle teenuste reklaamil. Interaktiivseid materjale seal ei pakuta.

3.4 Enesearengu päevikud

Arengupartneri ja organisatsioonide koolitaja Signe Ventseli sulest on ilmunud mitmed enesearengu töövihikud, nagu "100 päeva fookus" ja "Ole iseenda boss", mille missioon on kalendrivormis märkmikusse eesmärgid seada ning nende täitmist teatud perioodi vältel üles märkida. Loodud lahendus ei viita aga teaduspõhistele uuringutele, ei suuna teraapiasse ega koolitustele. Sisu on mõeldud olema inspireeriv, leidub rohkelt tsitaate avaliku elu tegelastelt. Rõhutatakse sisekaemuse tähtsust, motivatsiooni hoidmist, regulaarsust, tagasivaadet tehtule.

3.5 Inglisekeelsed rakendused

Inglise keeles on loodud on ka mitmeid rakendusi, aitamaks neid, kes kannatavad depressiooni, ärevushäirete või post-traumaatilise stresshäire (PTSH) all ning ka rakendusi, mis toetavad vaimse tervist, püüdes suurendada keskendumisvõimet ja rahulolu teadveloleku ja meditatsiooni abil [20], näiteks Headspace, Insight Timer, Smiling Mind, Inscape, Calm ja Streaks. Kuna vaadeldav töö keskendub eelkõige vaimse tervise edendamisele ega hõlma vaimse tervise häireid, mis võivad vajada kliinilist sekkumist, siis on järgmisena välja toodud just näited mõnedest eelnimetatud vaimset tervist toetavatest rakendustest.

3.5.1 Headspace

Positiivse vaimse tervise rakendus Headspace sisaldab animeeritud meditatsiooniharjutusi, probleemide teadvustamist soodustavaid õppevideoid, et suurendada teadvelolekut. Sarnaselt teistele teadvelolekule ja meditatsioonile keskendunud rakendustele ei paku Headspace võimalust oma isiklikku arengut monitoorida ja ülestähendusi pidada ning on vaid lühikese prooviaja jooksul tasuta.

3.5.2 Streaks

Üldisema monitoorimise rakendusena on loodud Apple disainiauhinnaga pärjatud rakendus Streaks, kus saab ise määrata ükskõik, millise harjumuse (suitsetamise vältimisest koeraga jalutamiseni välja) ning selle täitmist ise jälgida. Rakendus on väga jõulise kasutajaliidesega ning toimib koos muude terviseandmeid monitoorivate rakendustega, sh nutikelladega. Vaadeldava töö lähtepunktist hinnates võib miinusena välja tuua, et rakenduse rõhuasetus pole vaimsel tervisel ning rakenduse kasutamine on tasuline.

Kokkuvõte eri vaimset tervist toetavatest digilahendustest on toodud välja tabelis 1.

Tabel 1. Olemasolevate lahenduste võrdlustabel.

	Portaal Terviseinfo	Portaal Peaasi	Portaal Vaimne tervis	Päevik "100 päeva fookus"	Päevik "Happy Journal"	Rakendus Headspace
Erialaselt usaldusväärne	x	x	x			?
Teadlikkuse suurendamine	x	x	x			
Nõustamisele suunamine		x	x			
Testimine		x				x
Teraapia-materjalid						
Koolitused		x				x
Tasuta	x	x	x			
Erinevad eagrupid	x	x	x			
Töövahend		x		x	x	x
Harjumuste monitoorimine				x	x	x
Päevik				x	x	
Interaktiivne						x

Olemasolevate lahenduste analüüsist selgus, et teadaolevalt puuduvad võimalused toetada vaimset tervist interaktiivsel moel erialaselt usaldusväärsete meetoditega ja eesti keeles nii iseseisvalt kui ka teraapia käigus. Sellest lähtuvalt on käesoleva töö eesmärk luua eelpool nimetatud võimalustega vaimse tervise toetamise veebirakenduse prototüüp.

3.6 Loodava lahenduse skoop

Loodava rakenduse puhul saab olema tegu veebirakenduse MVP lahenduse prototüübiga, mida on edasi arendades võimalik avalikult kättesaadavaks teha. Loodav rakendus võimaldab kasutajal sisse logida, teha eneseanalüüsi vormide sissekandeid, et enda vaimse tervise näitajaid monitoorida. Plaan on töötada välja terviklik lahendus ühe eneseanalüüsi vormi põhjal, mille tulemusel saab seda vormi korduvalt täita ja varasemaid sissekandeid üle vaadata. Vormide loomisel arvestatakse teaduspõhistele allikatele tuginevaid spetsialistide soovitusi eri eestikeelsetelt vaimse tervise lehtedelt ning kooskõlastatakse muid lahendusi vaimse tervise spetsialistiga, kelle hinnangut rakenduse loomisel kaasatakse.

Samuti saab rakenduse kaudu psühholoogiga koostööd teha, et neid eneseanalüüsi vorme jagada ja hallata, et teraapiaprotsessi hõlbustada.

Loodav rakendus toetab eri keelevalikute lisamist, plaan on luua keelevalik vähemalt inglise ja eesti keele jaoks, vähendades eesti keelt kõnelevate kasutajate jaoks keelebarjäärist tulenevaid takistusi. Samuti on eesmärk pakkuda tasuta baasversiooni, võimaldades tööga kohe algust teha ning piiramata aja jooksul põhifunktsioone edasi kasutada.

Käesoleva lõputöö skoobi väliselt on võimalik luua tasuline täisversioon, mis sisaldab edasiarendatud võimalusi, mille arendusel oleks vaja kaasata lisaressursse rakenduse arendamise, evitamise ja haldamise jaoks.

4 Rakenduse IntroScope analüüs

Rakenduse IntroScope prototüübi analüüsis vaadeldakse lähemalt, millised saavad olema loodava rakenduse nõuded, millest lähtutakse kasutajakogemuse disainimisel ning milliste tehnoloogiate abil oleks võimalik rakenduse tarkvara arendada.

4.1 Kasutajakogemuse disain

Iga veebirakenduse arendamisel, kus on lõppkasutajaks inimene, on lahutamatuks osaks kasutajakogemuse disain. See on protsess, mille käigus mõeldakse läbi, kuidas inimene ja masin omavahel suhtlevad. Eriti oluline on see infosüsteemide puhul, mis vahendavad inimesele eluks vajalikke teenuseid või on suunatud haavatavas olukorras kasutajale. Kuna vaimse tervise rakenduse potentsiaalsed kasutajad võivad liigituda just viimasesse gruppi, tuleb vaimset tervist toetava rakenduse loomisel kasutajakogemuse disain hoolikalt läbi mõelda.

Üheks oluliseks allikaks, mida kasutajakogemuse kujundamisel arvesse võtta, on WCAG (Web Content Accessibility Guidelines) ehk veebisisu juurdepääsetavuse juhised. Need koostas organisatsioon W3C (*World Wide Web Consortium)*, mis on põhiline rahvusvaheline internetistandardite organisatsioon. WCAG kohaselt peab veebisisu vastama neljale põhiomadusele [21]:

1. **Hoomatavus** (*perceivability*) – informatsioon ja kasutajaliidese komponendid peavad olema esitatud kasutajale hoomataval viisil. See tähendab näiteks, et mittetekstilist sisu peab olema võimalik muuta ka teksti kujule. Veebirakenduses on see näiteks lahendatav viisil, kus piltide HTML (*HyperText Markup Language*) elemendile lisatakse juurde atribuut *alt*, kuhu lisatakse pildi kirjeldus. Sel moel saavad näiteks vaegnägijad ekraanilugeri abil paremini aimu, millist infot veebileht sisaldab. Samuti on mõeldud hoomatavuse all muid inimese taju arvestavaid omadusi (värvi-lahendused, kontrastsus, esituskiirus, pealkirjade lisamine), tagamaks, et veebisisu oleks esitatud kasutajale arusaadavas vormis. Värvide kontrastsus peab vastavalt

standardile WCAG 2.0 olema normaalse teksti puhul vähemalt 4.5:1 ning suure teksti puhul 3:1. Värvide kontrastsust saab määrata vabavaraliste veebirakenduste abil, nagu näiteks Contrast Checker [22].

2. **Käsitletavus** (*operability*) – selle põhimõtte kohaselt peavad kasutajaliidese komponendid ja navigatsioon olema lihtsasti käsitletavad. Näiteks peab kasutajal olema tegevuste jaoks piisavalt aega, funktsioonidele peab saama ligi klaviatuuri abil ning kasutaja peab saama veebikeskkonnas navigeerida, sisu leida ja oma asukohta kindlaks teha.
3. **Mõistetavus** (*understandability*) – teksti kujul sisu peab olema loetav ja mõistetav. See tähendab näiteks, et rakenduses kuvatavad veateated peavad olema inimloetaval kujul, veebilehed peavad avanema ning toimima oodatud viisil, kasutaja peab saama vigu vältida või parandada. Selle muudavad hõlpsamaks informatiivsed veateated, mis märgivad ära koha, mida parandama peaks.
4. **Töökindlus** (*robustness*) – selle põhimõtte järgi peab veebisisu olema võimalikult ühilduv kasutajaagentidega (*user agent*), näiteks eri brauseritega. Selleks tuleb kasutada korrektselt HTML-i ehk veebisisu loomise üht peamist tehnoloogiat. See tähendab, et HTML-i sildid (*tags*) peavad olema korrektse struktuuriga, näiteks kasutama unikaalseid identifikaatoreid (atribuute nimega *id*) HTML dokumendi siseselt vaid ühe korra.

Kasutajakogemuse guruks tituleeritud [23] Jakob Nielseni sõnul on halva kasutajakogemuse põhjusteks pahatahtlikkus, rumalus või laiskus [24]. Näiteks võib tema sõnul pidada pahatahtlikuks disainiks juhtumit, kus infosüsteemis on teadlikult raskendatud kliendil teenusest loobumine, et ettevõte saaks jätkuvalt teenuse eest raha küsida. Rumala disaini põhjusena toob ta välja teadmatuse ning selle lahendamiseks teadlikkuse tõstmise tüüpilistest kasutajakogemuse disaini vigadest. Viimasena kirjeldab Nielsen laiska disaini, mis ei tulene teadmatusest, vaid nagu nimigi ütleb, laiskusest või siis pelgalt UX-i (*User Experience* ehk kasutajakogemus) madalast prioriteedist piiratud ressurssidega projektis. Selle korrigeerimiseks on samuti abi teadlikkuse tõstmisest sellest, mis kasutajaid häirib või nende huve kahjustab. Ta loetleb kümme peamist viga, mida aastal 2021 kõige häirivamaks on peetud:

1. **Hüpikaknad ja pealised** (*pop-ups and overlays*) – UX-i mõttes kaetakse mingi vajalik osa kasutaja vaatest ära. Kui juba üks hüpikaken on halb, siis paljude kasutamine korraga on lausa kohutav. Hüpikaknaks võib olla reklaam, luba küpsiseid

kasutada, vestlusaken või küsitlus. Kasutaja pannakse olukorda, kus ta peab enne plaanipärase tegevuse sooritamist töötama läbi kaasneva inforägastiku ja aknad ükshaaval sulgema. Kasutaja eelistaks rakenduselt rahulikku keskkonda ja vaikust liigsest infomürast.

2. **Aeglane reageerimisaeg** – probleemiks võib olla lehe kui terviku laadimine. Tänapäeval kasutatakse küll lairiba, kuid seda pole kõigil. Arvestada tuleks ka maapiirkondadega ja kasutajatega, kelle telefon lülitub tühjenedes energiasäästu režiimile. Laadimise kiirust mõjutab tihtipeale asjaolu, et süsteemi eri komponendid laetakse alla eri pilveserveritest. Ideaalis võiks leht olla laetud vähem kui sekundiga. Kui laadimine on aeglasem, siis kasutajale tundub, et ta peab arvuti järel ootama. Seega navigeerimisvabadus väheneb, niisiis ka navigeeritakse vähem, mille tulemusena rakenduse kasutatavus väheneb.
3. **Petlikud lingid** – kasutaja jaoks on link lubadus, et sinna klõpsates saab kuhugi minna. Kui see ei toimi, siis veetakse kasutajat alt. Samuti tuleks kasutaja ootusi juhtida õigesti selles osas, mis täpsemalt lingile klõpsates avaneb: on see video, esitlus, artikkel või registreerimisvorm?
4. **Väike tekst madala kontrastiga taustal** – probleem peamiselt vanemaealistele, kuid mitte ainult. Kasutaja ei peaks olema sunnitud info lugemiseks lisapingutusi tegema, näiteks lähemale naalduma või silmi kissitama.
5. **Paindumatu sisendi vorming** – kui süsteem kehtestab, et infot peab sisestama väga kindlal viisil. Näiteks tahetakse, et krediitkaardi number oleks jaotatud neljasteks numbriplokkideks ega lase kasutajal infot sisendväljale muul kujul kleepida. Kasutajale peaks jääma tunne, et tema ohjab olukorda. Programm peaks ise tühikud, sidekriipsud vms eemaldama või sobiliku vormingu kasutajale ette näitama. Paindumatu sisendi eeldamine on kasutajale häiriv ning muudab ka äriloogilise protsessi veaaltimaks.
6. **Põhivõimaluste keelamine** – kui keelatakse ära sisu valimine, kopeerimine ja kleepimine. Kasutaja ei saa sisu jagada ning selle tulemusena võib infosüsteem ka rahaliselt kaotusi kanda.
7. **Nimetud ikoonid** – ainult pildi järgi on raske aru saada, mida ikooniga nupp teeb. Näiteks südameikoon ühel lehel võib tähendada hoopis muud teisel lehel. Teadmatus, mida nupp täpsemalt teeb, tekitab kasutajale ärevust ning seetõttu ei pruugi ta nuppu kasutada. Kõige lihtsam oleks lisada ikoonile juurde ka tekst selle tähenduse kohta.

8. **Veebilehe laadimisel paigutuse nihkumine** – see on esteetiliselt häiriv, kuna taustal toimub pidev liikumine. Kasutajad on kannatamatud, nad tahavad kohe „asja kallale asuda“. Võib tekkida olukord, kus kasutaja näeb midagi, mida soovib avada, kuid klõpsamise hetkel on see element paigast nihkunud ning kasutaja klõpsab lõpuks mingi muu lingi peale ja satub ootamatusse kohta.
9. **Tohutu suur pilt avakuval** – kui see on veebilehe peamine pilgupüüdja, võib see tekitada tunde, et leht on lõplik. See tähendab, et kasutajale ei jää visuaalset muljet, et lehe sisu kuskil jätkuks (näiteks alla kerides). Leht peaks kasutajale kommunikeerima, mida seal teha saab. Seal peaks olema palju valikuid.
10. **Mobiilidisain suurtel ekraanidel** – arendajad eeldavad tihtipeale, et mobiilivaadet võib lihtsa vaevaga suurendades ka suuremal ekraanil kasutada. Näiteks burgermenüüd, kus osa tähtsat infot on kokku pakitud, kuigi suurel ekraanil mahuks seda ka avatuna kuvama. Kasutajad ei kipu neid peidetud valikuid sellistel puhkudel üldse avama. Seega vajavad suured ekraanid eraldi läbi mõeldud disaini lahendusi.

Kõigi loetletud disainivigade ühise joonena toob Nielsen välja üleolevust – olukorda, kui veebikeskkond peab ennast kasutajaks tähtsamaks ning soovib kasutaja tegevust dikteerida.

Kui Nielsen keskendus kasutajakogemuse vigadele, siis Ben Shneiderman, inimese ja masina vahelise suhtluse ekspert, on sõnastanud kaheksa kuldreeglit, mida UI disainerid peaksid kasutajasõbralikkuse saavutamiseks silmas pidama [25]:

1. **Taotle järjepidevust** – kogu vaate kujundus (elementide paigutus, nuppude suurus, värvipalett) peaks olema ühtlane ning aitama kujundada süsteemi identiteeti.
2. **Paku sagedastele kasutajatele otseteid** – kasutajad peaksid saama kõikjale minimaalsete klõpsude arvuga. Selleks peab olema menüül selge ja läbimõeldud hierarhia.
3. **Anna asjakohast tagasisidet** – veebilehel toimetav kasutaja peaks saama kohest tagasisidet oma protsessi hetkeseisu kohta. Sellist tagasisidet annavad näiteks laadimise ja katkestamise nupud ja teated.
4. **Kuva lõpetatuse dialooge** – iga suhtlus peaks saama ka loogilise lõpu, vähendamaks kasutaja mentaalset koormust.
5. **Paku lihtsat veahaldust** – heas kasutajaliideses välditakse vigu nii palju kui võimalik. Ent kui midagi siiski juhtub, peaks süsteem kasutajale inimkeeli mõista

andma, kuidas probleemi lahendada. Seda saab teha näiteks selgete veateadete ja lahendusvõimaluste kuvamisega.

6. **Võimalda lihtsasti tegevus tagasi võtta** – kasutaja tunneb end turvalisemalt, kui tal on võimalik probleem lahendada „Tagasi" või „Tühista" nupu abil. Sel juhul julgeb ta vabamalt eri võimalusi uurida.
7. **Toeta seesmist kontrollikeset** – kasutajal peaks olema vabadus süsteemi ise kontrollida ja otsuseid teha. Näiteks süsteemi seadeid teatud määral muuta, et reguleerida, millist infot neile saadetakse või kuvatakse.
8. **Vähenda koormust lühimälule** – inimese lühimälu maht on piiratud, seega peaksid kuvasätted olema võimalikult minimalistlikud. Infopaigutust tuleb prioriseerida, et kasutaja tähelepanu kõige olulisemale suunata ning vältida liigset infomüra.

Vaimse tervisega seotud rakenduse puhul on nende eelpool toodud soovituste taustal vaja UX kindlasti kõrgendatud tähelepanu alla võtta. Seda põhjusel, et tegu on potentsiaalselt haavatavama sihtgrupiga ning rakendusest peab kujunema kasutaja jaoks usalduslik keskkond. Selle tagamiseks tuleb võtta arvesse WCAG soovitusi ning püüda vältida põhilisi UX disaini vigu. Käesoleva töö raames ei mahu kõikide eelpool mainitud nõuannete rakendamine küll töö skoopi, kuid peatükis 4.2.2 on kõik planeeritavad mittefunktsionaalsed nõuded üles loetletud ning ülejäänute analüüs ning rakendamine jääkvad edasiarenduste raamesse.

4.2 Rakenduse nõuded

Käesoleva töö raames loodava rakenduse IntroScope prototüübi skoobi määramisel võeti arvesse eelkõige olemasolevate rakenduste põhjal kogutud ideid, eriala spetsialisti soovitusi ning kasutajakogemuse disaini põhimõtteid. Saadud info põhjal pandi kokku kasutajalood, mis koondati ühisosa järgi eepostesse.

Rakenduse kasutajad jagunevad esialgu kolme rolli: rakenduse administraator, kasutaja ja nõustaja. Administraator sisestab eneseanalüüsi vormid rakendusse. Põhilisi eneseanalüüsi vorme saab kasutada iga rakendusse sisse loginud kasutaja. Juhul kui rakendust kasutatakse teraapiaprotsessis, lisab kasutaja rakenduse kaudu terapeudi, kes omakorda ligipääsu võimaluse kinnitab. Klient saab määrata, millistele vormidele ta nõustajale ligipääsu annab.

4.2.1 Funktsionaalsed nõuded

Järgmisena on tabelites 2 ja 3 on toodud välja rakenduse funktsionaalsed nõuded, mis kirjeldavad infosüsteemi põhitegevusi lõpp-kasutaja jaoks [26]. Need on esitatud kasutajalugude kujul, mis omakorda on grupeeritud ühiste omaduste alusel eeposteks. Tabelis 2 on toodud rakenduse nõustaja eeposed ja kasutajalood.

Tabel 2. Rakenduse nõustaja eeposed ja kasutajalood.

ID	Roll	Tegevus	Eesmärk
4	**Nõustaja**	peab saama temaga jagatud vorme hallata,	et teraapiat tõhusamalt korraldada.
4.1.	Nõustaja	peab saama ülevaate kõikidest, kes temaga on vorme jaganud,	et klientide andmeid paremini hallata.
4.2.		peab saama näha kõiki konkreetse kasutaja jagatud vorme kategooriate kaupa,	et valmistuda teraapiaks olenevalt teemast.
4.3.		peab saama näha konkreetse kasutaja jagatud vorme vormi liigi kaupa,	et määratud kodutöid enne teraapiat üle vaadata.
4.4.		peab saama näha vormi detailvaadet,	et saada täidetud vormist parem ülevaade.
5	**Nõustaja**	peab saama koostöökutseid hallata,	et kontrollida, milliste klientidega koostööd teha.
5.1.	Nõustaja	peab saama kinnitada kliendi uue koostöökutse,	tagamaks, et koostöövalmidus on mõlemapoolne.
5.2.		peab saama jagatud vormi vaatamisest keelduda,	kui koostöö ei ole varem kokku lepitud.
5.3.		peab saama teise kasutaja vormide saatmise blokeerida,	kui klient saadab korduvalt soovimatuid ligipääsutaotlusi.

Tabelis 3 on toodud välja rakenduse tavakasutaja eeposed ja kasutajalood. Kuna iga nõustaja on ühtlasi ka tavakasutaja rollis, siis kehtivad kõikidele nõustajatele ka tavakasutajate võimalused. Nii saab nõustaja ka ise vormidega tutvuda, kuid talle kui nõustajale laienevad õigused teha rakenduse kaudu klientidega koostööd nõustamise toetamiseks.

Tabel 3. Rakenduse tavakasutaja eeposed ja kasutajalood.

ID	Roll	Tegevus	Eesmärk
1	**Kasutaja**	tahab täita privaatselt eneseanalüüsi vorme,	et enda vaimset tervist toetada.
1.1.	**Kasutaja**	peab saama sisse logida,	et tagada diskreetne sisuloome.
1.2.		peab saama näha kõiki eneseanalüüsi kategooriaid,	et saada ülevaade rakenduse võimalustest.
1.3.		peab saama näha valitud kategooria eneseanalüüsi vorme,	et saada ülevaade kategooria võimalikest vormidest.
1.4.		peab saama ligi enda varem täidetud vormidele,	et saada ülevaade enda arengust.
2	**Kasutaja**	tahab enda täidetud vorme oma nõustajaga jagada,	et muuta teraapiaprotsess tõhusamaks.
2.1.	**Kasutaja**	peab saama iga vormi eraldi nõustajaga jagada,	et kontrollida, milliseid vorme nõustaja näeb.
2.2.		peab saama ülevaate, milliseid vorme ta on kellega jaganud,	et vormide ligipääsu hallata.
2.3.		peab saama nõustaja ligipääsu vormile eemaldada,	et vormide ligipääsu hallata.
2.4.		peab saama määrata vormi ligipääsule tähtaja,	et piirata ligipääsu aega.
3	**Kasutaja**	peab saama rakendusest asjakohast lisainfot,	et saada teadlikumaks vaimset tervist puudutavatest teemadest.
3.1.	**Kasutaja**	peab saama lingi abil infot kohese abi allikatest,	et vajadusel kriisiabi saada.
3.2.		peab nägema lisainfo materjale,	et huvi korral rohkem lugeda.
3.3.		peab saama eneseanalüüsi vormidega koos täiendavat infot,	et saada parem ülevaade vormi eesmärgist.

4.2.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded kirjeldavad infosüsteemi kvaliteeti [26]. Mittefunktsionaalsete nõuete loomisel võetakse arvesse peatükis 3.1 toodud WCAG põhimõtted ja muud peatükis 4.1 välja toodud UX disaini põhimõtted, arvestades seejuures ka vaatluse all oleva lõputöö ajalist skoopi. Sellest lähtuvalt on loodava rakenduse mittefunktsionaalsed nõuded järgmised:

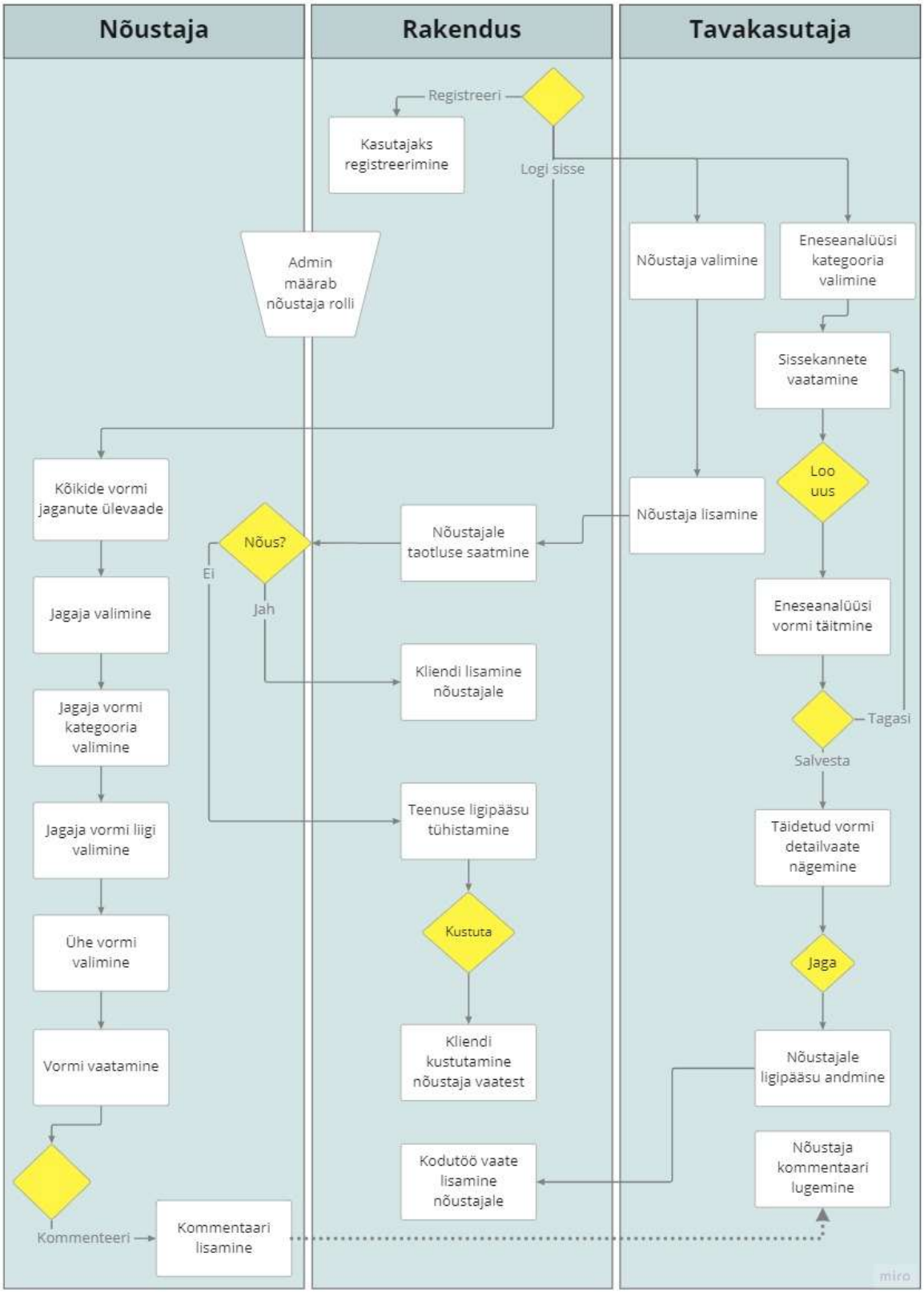
- Kasutaja peab saama rakenduse keelt vahetada.
- Rakendus peab võimaldama kuvada infomullidena selgitavaid tekste.
- Kasutajale peab kuvatama enne POST-päringuga seotud toiminguid kinnitusküsimusega modaali.
- Piltidele peab olema lisatud atribuut *alt* koos pilti kirjeldava tekstiga.
- Rakenduse värvilahendused peavad olema kontrastsed (kontrastsussuhe vähemalt 4.5:1).
- Rakendus peab kuvama kasutajale vajadusel mõistetavaid veateateid.

4.2.3 Kasutusvoo diagramm

Kasutajate tegevusi rakenduses ning rollide koostoimimist iseloomustab kasutusvoo diagramm, kus standardse modelleerimiskeele UML (*Unified Modeling Language*) abil on märgitud rakenduses toimuvad protsessid ja tegevused. Eri rollide interaktsioone väljendavad tulbad.

Loodavas rakenduses on esindatud kolm rolli: tavakasutaja, nõustaja ja rakenduse administraator. Rakendusse registreerides on iga kasutaja tavakasutaja rollid. Nõustaja rolli taotlemiseks tuleb kontakteeruda rakenduse administraatoriga ning esitada kutsetunnistust tõendav dokument. Seejärel määrab rakenduse administraator isiku nõustaja rolli. Rakenduse tavakasutajad saavad rakenduse kaudu taotleda nõustaja rollis olevatelt kasutajatelt nõustamist ning misjärel nõustaja kas kinnitab või tühistab taotluse. Nõustajad saavad seeläbi suunata oma kliente rakendust kasutama ning omavahel koostööd teha. Nõustamisel osalevad kliendid või tavakasutajad saavad täita eneseanalüüsi vorme ka ilma jagamata. Jagamine toimub iga täidetud vormi kohta eraldi.

Kasutusvoo diagramm eri rollide protsesside ja tegevuste kohta on toodud joonisel 1.



Joonis 1. Kasutusvoo diagramm.

4.3 Tehnoloogiavalik

Võimalusi veebirakenduse loomiseks on mitmeid. Järgmisena käsitletakse eri meetodeid, pidades silmas eelnevalt kirjeldatud probleemi ja selle lahendamiseks kogutud nõudeid. Samuti arvestatakse töö autori varasemate kogemuste ning lahenduse loomiseks kuluvate võimalike ressurssidega. Analüüsitakse vaid neid tehnoloogiaid, mille arendusprotsess kestab ning mis pakuvad kasutajatele pikaajalist tuge.

4.3.1 Programmeerimiskeel ja raamistik

Tarkvara arenduseks kasutatav raamistik on tihedalt seotud programmeerimiskeelega. Seega on raamistiku valikul oluliseks argumendiks oskus selle raamistiku programmeerimiskeelt kasutada. Tallinna Tehnikaülikooli IT-süsteemide arenduse õppekavas õpiti kasutama järgmisi tuntud serveripoolseid programmeerimiskeeli ning raamistikke:

C# ja .NET

Töö autoril on kõige põhjalikum kogemus tervikliku rakenduse arendamisel .NET raamistiku ja sellest lähtuvalt C# keelega ning ASP.NET raamistikuga, mis on mõeldud .NET-i ja C# abil veebirakenduste ja teenuste loomiseks [27]. Nende töövahendite abil on mugav luua tuntud veebimustri mudel-vaade-kontroller (*model-view-controller,* MVC) arhitektuurimustriga veebirakendusi. Tegu on levinud arhitektuurimustriga, mis jaotab rakenduse kolmeks loogiliseks komponendiks: mudel, vaade ja kontroller, mis on loodud tegelema rakendusespetsiifiliste aspektidega [28]. Kokkuvõtvalt võib öelda, et mudel tegeleb andmetega, vaade kasutajaliidesega ning kontroller andmete edastamisega vaatele [29]. C# kui programmeerimiskeel on tugevalt tüübitud ja objekt-orienteeritud (OO) keel [29], mis võimaldab luua programmi selge struktuuri järgi, vähendab koodikordusi, lihtsustab koodi haldamist, muutmist ja vigade otsimist ning muudab selle lihtsamini korduvalt kasutatavaks [30][31].

Python ja Django

Python on interpreteeritud, interaktiivne, objekt-orienteeritud programmeerimiskeel [32]. Väga lihtsa süntaksi taga on lai ampluaa eri võimalusi, teeke ja programmeerimise paradigmasid [32]. Üks populaarne raamistik, millega Pythoni keeles MVC veebirakendusi luua, on Django [33]. Kahjuks on kokkupuude selle raamistikuga ülikooli

Pythoni kursuse jooksul väga põgus ning veebirakenduse arendamisele Django abil peaks eelnema põhjalik lisatöö.

Java Spring

Teine tugevalt-tüübitud objekt-orienteeritud veebirakenduste arendamise töövahend on programmeerimiskeel Java ja seda kasutav Springi raamistik. Java pakub palju eri teeke, mida saab jooksvalt projekti lisada. Samas puudub töö autoril kogemus Java abil esirakenduse loomisest ning autentimise ja sisselogimise kasutajaliidese loomisest. Nende lahenduste juurde õppimiseks kuluks vähemalt nädal-paar lisaaega.

PHP ja Drupal

Veebirakenduse loomisel on väga tugev põhi ka Drupalil ning programmeerimiskeelel PHP, mida autor kasutab igapäevatöös. Drupal on üks levinumaid veebihalduskeskkondi Wordpressi kõrval. Paraku puudub autoril kogemus nende töövahendite abil rakenduse loomisel algusest lõpuni ning Drupali seadistamise õppimisele tuleks eraldi aega pühendada. PHP pole tugevalt tüübitud keel ning valede andmetüüpide kasutamisel pole kompilaatori tuge veakohtade leidmisel.

JavaScript ja Node.js

Node.js on vabavaraline platvormist sõltumatu keskkond serveripoolsete rakenduste loomiseks JavaScripti abil [33]. Autor on küll läbinud JavaScripti kursuse, kuid keskendunud pigem esirakenduste loomisele. Ajakulu täiendavate kursuste läbimiseks, et õppida Node.js-i abil tagarakendusi looma, oleks arvestatav. Samuti on JavaScript küll kergesti õpitav, kuid nõrgalt tüübitud programmeerimiskeelena võib suure rakenduse loomisel osutuda takistuseks. Mõnevõrra on abiks TypeScripti rakendamine, mis on tugevalt tüübitud ja JavaScripti peale ehitatud programmeerimiskeel, võimaldades koodiredaktori abil varakult vigu üles leida ning rakenduse mastaapi suurendada [34].

Tabelis 3 on toodud kokkuvõte kogemusest eri programmeerimiskeeltega ning hinnanguline lisatöö, mis tuleks teha nende kordamisele ja lisaks õppimisele.

Tabel 4. Programmeerimiskeelte kogemuse kokkuvõte.

	Python	Java	C#	PHP	JavaScript
Varasem kogemus	Hea	Väga hea	Väga hea	Hea	Hea
Kordamise ajakulu	Suur	Suur	Väike	Väike	Keskmine
Õppimise ajakulu	Suur	Suur	Väike	Väike	Keskmine

Võttes arvesse lõputöö skoopi ning selle loomiseks kuluvat ajaressurssi, oleks põhjendatud valida arenduseks töövahendid, millega on pikem varasem kogemus ning mis ei eelda suuremahulist juurdeõppimist. Käesoleva töö tingimustes on kõige mõistlikum valida selleks keeleks C#, kuna olemasolevate kogemuste põhjal on realistlik seatud eesmärgid saavutada, ilma et kuluks märkimisväärselt aega oskuste täiendamisele.

Sellest lähtuvalt oleks kõige mõistlikum valida arenduse raamistikuks .NET. Seda põhjusel, et C# ja .NET lahenduse kogemus on olnud rakenduse loomisel kõige terviklikum, hõlmates nii serveri- kui ka kliendipoolse lahenduse loomist, kasutaja autentimise ja õiguste kontrolli kehtestamist ning rakenduse evitamist Dockeri ja Microsoft Azure-i abil, mis võimaldab rakenduse hiljem hõlpsasti ka avalikult kasutatavaks muuta.

4.3.2 MVC või API

Rakenduse arhitektuuri mõttes on oluline otsustada, kas kasutada andmete edastamisel paradigmal REST põhinevaid veebiteenuseid, et mitte olla sõltuv ühest rakendusest ning luua võrgupõhine rakendusprogrammiliides (Web API) või on soov teha funktsionaalsus kättesaadavaks ühe rakenduse siseselt mudel-vaade-kontroller-mustri abil (MVC) [35]. Viimane võimaldab säästa aega, mis kuluks funktsionaalsusele ligi pääsemiseks eraldi API loomisele. Kuna vaadeldava rakenduse jaoks on vajalik eelkõige suhtlus kasutajatega vaadete kaudu, mitte niivõrd andmete edastamine, oleks mõistlik kasutada ASP.NET MVC lahendusi, mitte ASP.NET Web API-t [36][37].

4.4 Lähtekoodi versioonihaldussüsteemi valik

Veebirakenduse arendamiseks on vaja tarkvara, mille abil lähtekoodi eri versioone hallata. Versioonihaldussüsteemid pakuvad võimaluse muudatusi jälgida, mitmel arendajal samaaegselt projekti kallal töötada, veebipõhise kasutajaliidese abil faili eri versioonidest visuaalne ülevaade saada, mitme arendaja samaaegse muudatuse tõttu tekkinud vastuolusid lahendada [38]. Tuntumad selletaolised tööriistad, mida töö autor on kasutanud, on GitHub, GitLab ja BitBucket [38]. Neist kõigil on eelpool loetletud võimalused olemas ning üksikisikuna on nende väikeses mahus kasutamine tasuta. Samuti on kõigil nii graafiline kui ka käsureapõhine liides, mille abil eri versioonide vahel navigeerida. Kuna ühelgi neist keskkondadest ei ole teiste ees mingeid kaalukaid eeliseid ega puudusi, valitakse käesoleva töö lähtekoodi halduseks GitHub, mille kasutamise kogemus on olnud kõige hiljutisem ja mahukam ning seega on ka seal pakutavate võimalustega kohanemise periood kõige lühem. Samuti võimaldab GitHub kasutajaliidese abil jagada infot arendaja ja projekti kohta – mis programmeerimiskeeli on eri projektides kasutatud, kui suure mahuga on arendaja neisse panustanud. See info on oluline näitaja tarkvara arendaja kohta värbajatele ja koostööpartneritele.

4.5 Integreeritud arenduskeskkonna valik

Rakenduse arendamisel kasutatav integreeritud arenduskeskkond (*Integrated Development Environment*, IDE) peaks toetama tööd valitud raamistiku ja programmeerimiskeelega, milleks on antud juhul C# ja .NET. Samuti peaks olema selle kasutamine juba eelnevalt tuttav, et võimaldada keskenduda tööle, mitte töövahendite tundmaõppimisele. Antud juhul on sellisteks keskkondadeks Visual Studio Code ja JetBrains Rider. Viimase valiku suureks eeliseks on töö autori rohkem kui aastane kogemus. Rideril on sisseehitatud kompilaator ReSharper, mis võimaldab vigu varakult tuvastada ning pakub võimalusi neid kiiresti parandada [39]. .NET 6 tarkvaraarenduse komplekt ehk SDK (*Software Development Kit)* võimaldab uue funktsionaalsusena ka väiksemaid muudatusi koheselt veebirakenduses näha, ilma et seda peaks taaskäivitama (*hot reload*) [39]. Rideri abil on lihtne vigu otsida (*debug*) ja NuGet paketihalduri abil tarkvarapakette lisada [40]. Puudusena võib välja tuua asjaolu, et Rideri kasutamine on üldiselt tasuline, kuid tudengitele on õpingute ajal siiski ligipääs tasuta.

Visual Studio Code on Microsofti vabavaraline laialt kasutatav lähtekoodi redaktor, millel on suur kasutajabaas ning kuhu on võimalik lisada laiendusi (*extensions*) [40]. Paraku on selle kasutajaliides autori kogemuse põhjal projektihalduse mõttes ebamugavam, kuna projekti failide puu taanded on halvemini eristatavad, raskendades ülevaate saamist failide paiknemisest.

4.6 Veebirakenduse arhitektuur

Käesoleva töö veebirakenduse arhitektuur ehitatakse üles SOLID-põhimõttel [41], mille kohaselt peaks tarkvara arendus lähtuma järgmisest viiest printsiibist:

S – *Single Responsibility principle* ehk ainuvastutuse põhimõte. Selle põhimõtte kohaselt peaks klassil või moodulil olema vaid üksainus põhjus, miks muutuda. Teisisõnu peaks olema klasside funktsioon rangelt piiritletud, nii et ühe muudatuse tegemine mitut eri äriloogilist protsessi häirima ei hakkaks.

O – *Open-Closed principle* ehk avatuse-suletuse põhimõte. Selle põhimõtte kohaselt peaks tarkvaraühikut saama vabalt laiendada, kuid mitte muuta, kuna see võib süsteemi teiste osade toimimist häirida.

L – *Liskov Substitution principle* ehk Liskovi asenduspõhimõte – programmi objekte peaksid olema asendatavad nende alamtüüpidega.

I — *Interface Segregation principle* ehk liideste eraldamise põhimõte – parem mitu spetsiifilist liidest kui üks üldkasutatav liides.

D – *Dependency Inversion principle* ehk pöördsõltuvuse põhimõtte kohaselt tuleks toetuda abstraktsioonidele, kõrgema taseme moodulid ei tohiks sõltuda madalama taseme moodulitest.

Nendest põhimõtetest lähtuvalt koosneb loodava veebirakenduse arhitektuur järgmistest kihtidest:

- **Domain** – andmemudeli kiht lihtsate äriloogikata andmeklasside koondamiseks.
- **DAL** – (Data Access Layer) andmevahetuse kiht suhtluse vahendamiseks äriloogika ja andmebaasi vahel.

- **BLL** – (Business Logic Layer) rakenduse äriloogika kiht, kus pannakse teenuste kaudu kokku veebirakenduste kontrollerite jaoks vajalikud andmed.
- **WebApp** – veebirakenduse kiht.

4.7 Analüüsi kokkuvõte

Analüüsi käigus tehti ülevaade loodava rakenduse prototüübi kasutajanõuetest, lahati kasutajakogemuse põhimõtteid, millest kavatsetakse lähtuda. Vaadeldi ka eri tehnilisi võimalusi rakenduse arendamiseks ning nende eeliseid ja puudusi. Tehniliste võimaluste analüüsist tulenevalt valiti rakenduse arendamise raamistikuks .NET programmeerimiskeelega C# ning ASP.NET raamistik, mis on mõeldud .NET-i põhiste veebirakenduste loomiseks. Plaanitav rakendus lähtuks MVC-mustrist. Veebirakenduse arenduskeskkonnaks valiti JetBrains Rider ning lähtekoodi versioonihalduseks GitHub koodibaas.

5 Veebirakenduse arendus

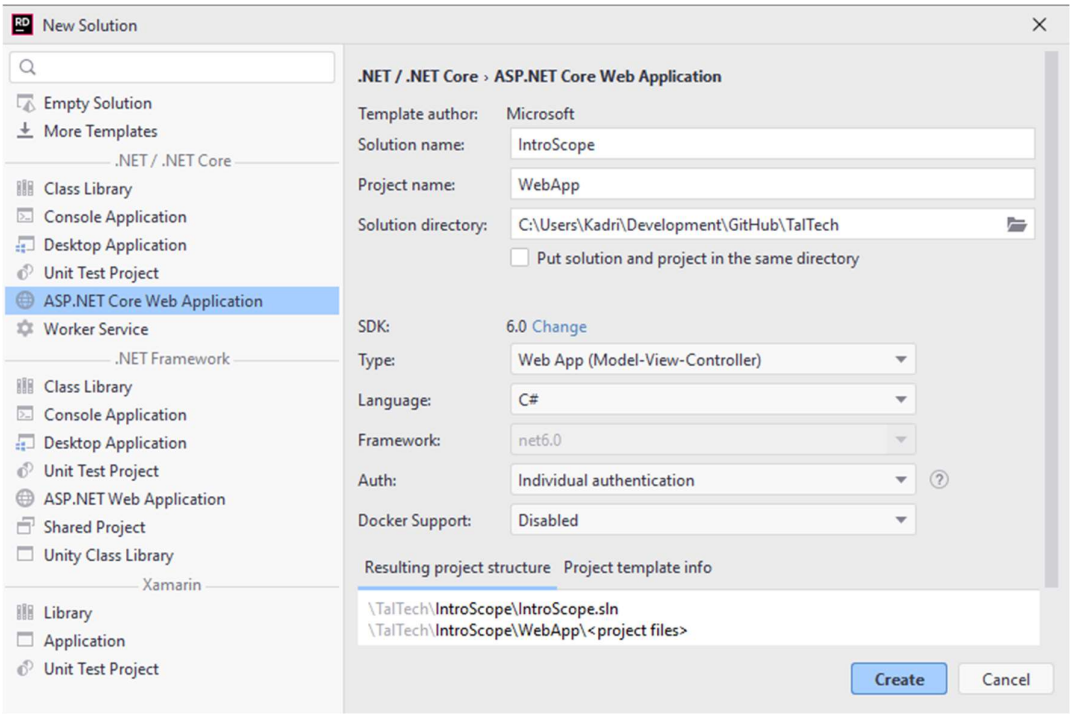
Vaadeldava rakenduse arendus jaguneb kolme etappi: rakenduse serveripoolne arendus, rakenduse kliendipoolne arendus, mis luuakse .NET raamistikul põhineva ASP.NET Core raamistikul. Lisaks rakenduse testimine. Iga etappi kirjeldatakse lähemalt eraldi peatükis.

5.1 Rakenduse serveripoolne arendus

Rakenduse serveripoolne lahendus põhineb .NET Core raamistikul ning koosneb eraldi andmetöötluse ja äriloogika kihist.

5.1.1 ASP.NET Core veebirakenduse loomine

Rakenduse loomine toimub IDEA UI kaudu, võimaldades valida eri tüüpi rakenduste vahel. Rakenduse projekti loomise kuva on lisatud joonisel 2.



Joonis 2. ASP.NET Core veebirakenduse loomine kasutajaliidese abil.

Antud juhul valitakse rakenduse tüübiks “ASP.NET Core Web Application”. SDK-ks valitakse .NET-i hetkel uusim versioon 6.0 ning autentimisviisiks “Individual authentication”, mis võimaldab luua spetsiifilisemaid õigusi ja võimalusi sisseloginud kasutajatele.

Vastloodud projekt sisaldab veebserverit Kestrel, mis on vaikimisi igasse ASP.NET Core projekti sisse ehitatud [42].

Projekti veebirakenduse kood on organiseeritud eraldi osadesse: vaated, kontrollerid, mudelid. Samuti on rakendusse seadistatud lihtne SQLite andmebaasimootoril põhinev testandmebaas arenduste jaoks, mis salvestab andmed otse kasutaja arvutisse. Kuna lõputöö eesmärgiks on luua veebirakenduse prototüüp ning kasutatakse testandmeid, siis sellest antud skoobiga arenduse jaoks täiesti piisab. Veebirakenduse serveripoolne kiht jaotatakse eraldi loogilistesse osadesse.

Domain

Projekti kaustas DOMAIN hoitakse rakenduse andmemudeleid POCO-sid (*Plain Old C# Object*). Need on lihtsad andmeobjektid, mida kasutatakse suhtluses andmebaasiga ning mille raames ei toimu äriloogilisi tegevusi [43]. Projektikaust DOMAIN sisaldab omakorda kahte projekti:

- **Domain.Base** – selles projektis on loodud abstraktne olemiklass BaseEntity, millest pärinevad kõik muud olemid ning millelt päritakse andmebaasi jaoks oluline primaarvõti.
- **Domain.App** – siin projektis on eraldi klassidena rakenduse äriloogika jaoks olulised olemid vajalike atribuutide ning olemite omavaheliste suhete kirjeldustega. Tänu .NET-i rakenduste raamistikule Entity Framework, mis tegeleb objekt-relatsioonvastendusega (*Object Relational Mapping* – ORM), saab tarkvaraarendaja keskenduda domeeniklassidele kui abstraktsioonidele, ega pea andmebaasiga suhtlemiseks kasutama rakenduse koodis otse andmebaasi päringukeeles SQL (*Structured Query Language*) päringuid, mis muudaksid koodi pikemaks ja raskemini hoomatavaks [44].

DAL

Rakenduse andmevahetus käib andmete ligipääsu kihi ehk DAL-i (*Data Access Layer*) kaudu. See koosneb omakorda projektidest DAL.Base ja DAL.App:

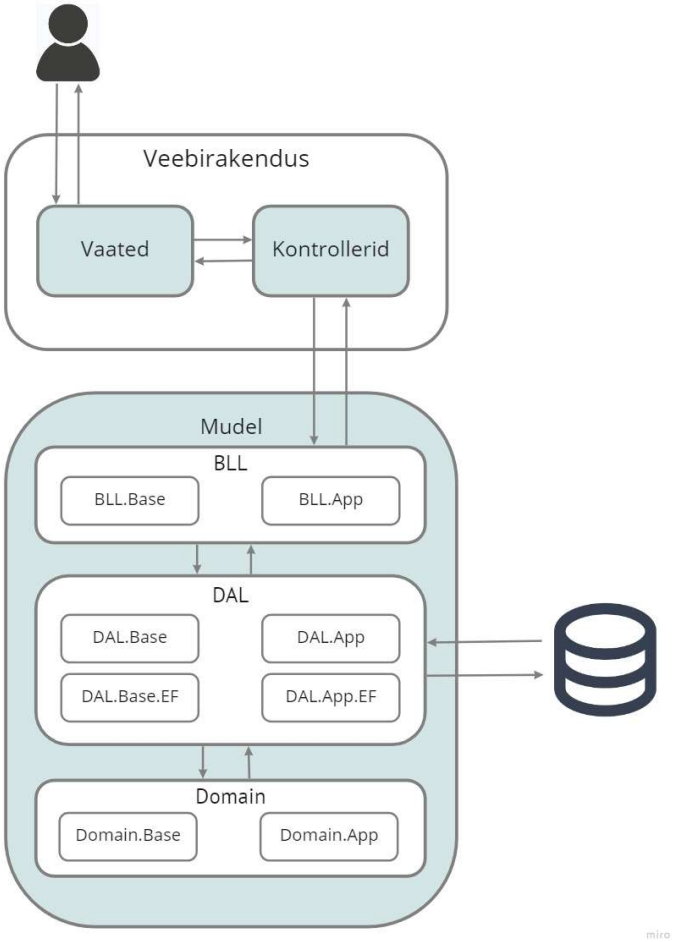
- **DAL.Base** – siin projektis asub andmeedastusobjektide ehk DTO-de (*Data Transfer Object*) vastendamise jaoks oluline klass BaseMapper, mis võimaldab .NET-i teegi AutoMapper abil eri projektikihtide DTO-sid vastendada (*mapping*), muutmaks andmemahukaid domeeniobjekte rakenduse muudes kihtides kasutamiseks sobivamaks.
- **DAL.Base.EF** – sisaldab klassi BaseRepository, mis kirjeldab repositooriumite üldist funktsionaalsust, mida kõik selle klassi alamklassid kasutavad.
- **DAL.App** – projekt hõlmab endas DTO-sid, vastendamisreegleid, mille abil andmebaasiobjekte DTO-deks teisendada ning andmebaasi loomise ning migreerimise jaoks vajalikke klasse.
- **DAL.App.EF** – siin on loodud eraldi repositooriumid, mis pärinevad klassist BaseRepository, kus toimub suhtlus iga olemi andmebaasitabeliga ning kus kirjeldatakse vajadusel spetsiifilisi funktsioone, mida BaseRepository ei paku.

BLL

Projektikaustas BLL (*Business Logic Layer*) asub rakenduse äriloogika kiht, mis on oluline vahekiht DAL-i ja vaadete vahel. Seal pannakse vastavalt äriloogikale kokku veebirakenduse kontrollerite jaoks vajalikud andmekogud, võimaldades kontrolleritel jääda koodipuhtaks ning äriloogikat üle kogu rakenduse taaskasutada [45].

- **BLL.Base** – projekt, kus asuvad baasklassid. Seal on näiteks BaseEntityService, mis on üldine teenusklass, kirjeldamaks teenustele omaseid funktsioone, mida alamklassidele pärandada.
- **BLL.App** – projekt, mis sisaldab DTO-de ja teenuste klasse. Teenused kasutavad eri repositooriume, et panna kokku vajalikke andmekogumeid, mida veebi-rakenduse vaadetele edastada. Näiteks on seal SurveyService, mille abil saab luua uusi eneseanalüüsi vorme, leida üles kõik konkreetse kasutaja vormid või kindlad vormiliigid.

Ülevaate veebirakenduse IntroScope MVC-mudelil põhinevast arhitektuurist on toodud joonisel 3.



Joonis 3. Rakenduse MVC-mudeli arhitekuurikihid.

5.1.2 Rakenduse andmebaasimudel

Rakenduse andmete hoidmiseks on loodud relatsiooniline andmebaasimudel programmiga Vertabelo. ERD-skeemil on toodud välja ülemklass Form, mis on üks-mitmele seoses klassidega Category ning eraldi moodulitena alamvormid. Kasutaja täidetud vormide haldamiseks mitu-mitmele seosega on loodud vahetabel FilledForms, kuhu jäädvustatakse kasutaja ning vormi primaarvõti, võimaldades samal kasutajal täita sama vormi korduvalt ning talletada selleks vajalikud andmed.

Esialgne kavatsus oli luua kogu andmestruktuur dünaamiliselt – iga vorm, selle juurde kuuluv küsimus ning vastus oleksid salvestatud eraldi andmebaasitabelitesse (vorm,

täidetud vorm, küsimus, vastus jne). Sellise mudeli abil aga tekkis vaadete loomisel tõrkeid.

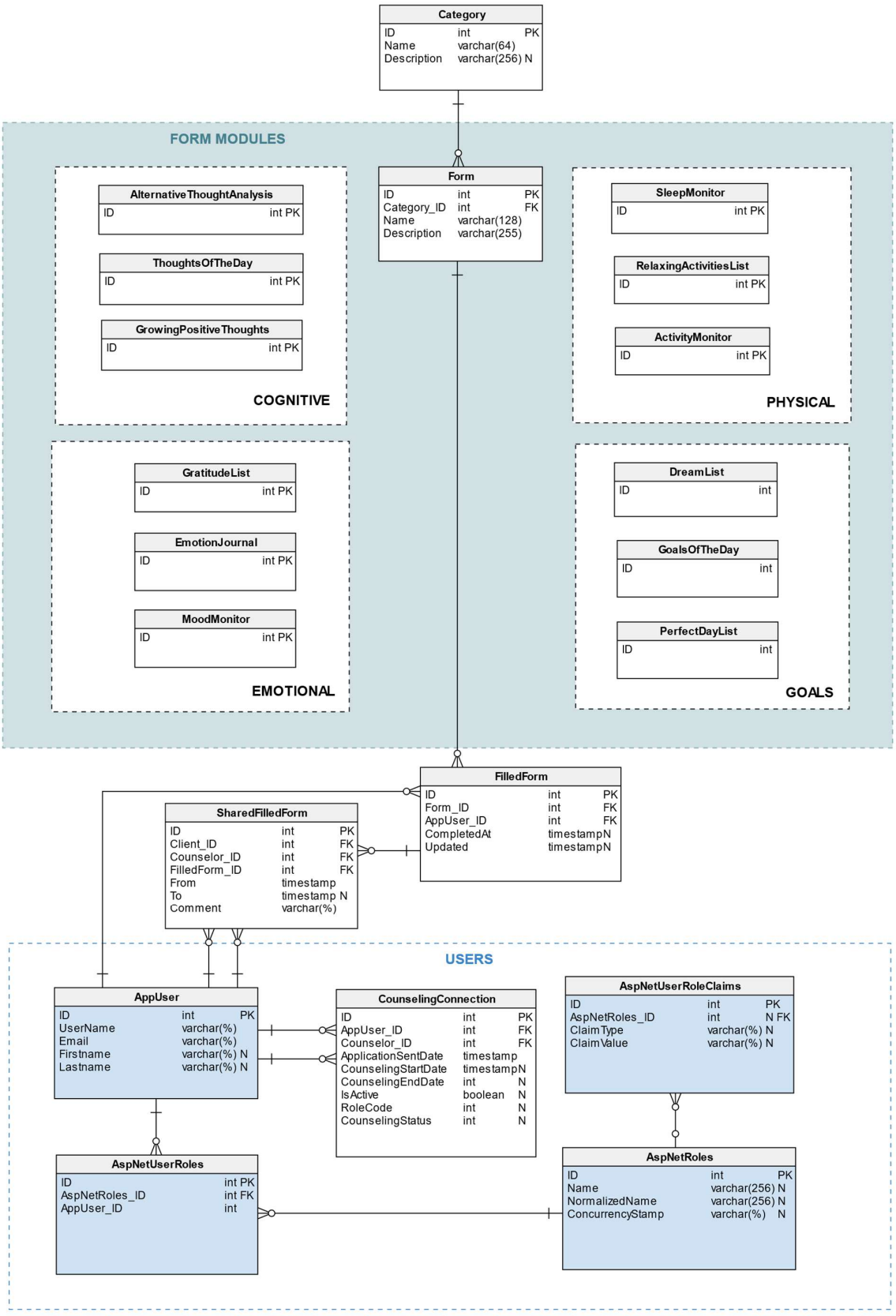
Kuna .NET kasutab vaadete genereerimisel tugevalt tüübitud andmeobjekte, siis muutus dünaamilise vormi loomine, kus küsimuste juurde kuuluvad sisendväljad luuakse vormi kuvamisel automaatselt, üsnagi keerukaks. Lisaks ei võimalda dünaamiline vormide esitamine kuigi paindlikult eri vorme neile omasel moel kujundada. Seega tundus mõistlikum kasutada iga vormitüübi jaoks eraldi andmeobjekti selle juurde kuuluva kontrolleri ja vaatega. Sel juhul on andmebaasitabelitest selgem ülevaade vormist kui tervikust ning andmete liikumine objektidena hõlpsamini hallatav. Kuna rakenduse vormide kogusumma ei tule tõenäoliselt kolossaalselt suur, ilmselt jääks see alla saja, siis pigem on mõistlik jaotada andmeid vormide kaupa, mitte killustada ühe vormi juurde kuuluvaid andmeid eri tabelite vahel.

Vormid jaotusid sõltuvalt eri vaimse tervise aspektist järgmistesse kategooriatesse: kognitiivne, emotsionaalne, füüsiline, eesmärkidega seonduv. Rakenduse arendamise käigus saab kategooriate ja vormide loetelu täiendada, jättes samuti paindlikkuse neid erinevalt kujundada.

Andmebaasimudelit koostades tekkis vajadus uurida teegi Entity Framework loogikat andmete vastendamisel, juhul kui kasutatakse abstraktseid ülemklasse, millelt selle alamklassid ühiseid andmeid pärivad. Näiteks on kõikidel andmeobjektidel primaarvõti, samuti kategooriat määrav välisvõti. Nende atribuutide igale alamvormile lisamine oleks läinud vastuollu tarkvara arenduse põhimõttega DRY (*Don´t Repeat Yourself*) [46], mille kohaselt tuleks vältida tarbetuid koodikordusi. Seetõttu tuli rakenduse andmebaasikonteksti lisada täiendavaid andmebaasi loomise reegleid. Samuti pidi ülemklassile märkima eraldi annotatsiooni [NotMapped], et sellest andmebaasitabelit ei tekitataks.

Rakendusesisese vormide jagamise loogika võimaldamiseks on tabel AppUser seotud tabeliga FilledForm kahekordse üks-mitmele seosega, et ühel kasutajal oleks ligipääs nii enda vormidele kui ka temaga jagatud vormidele.

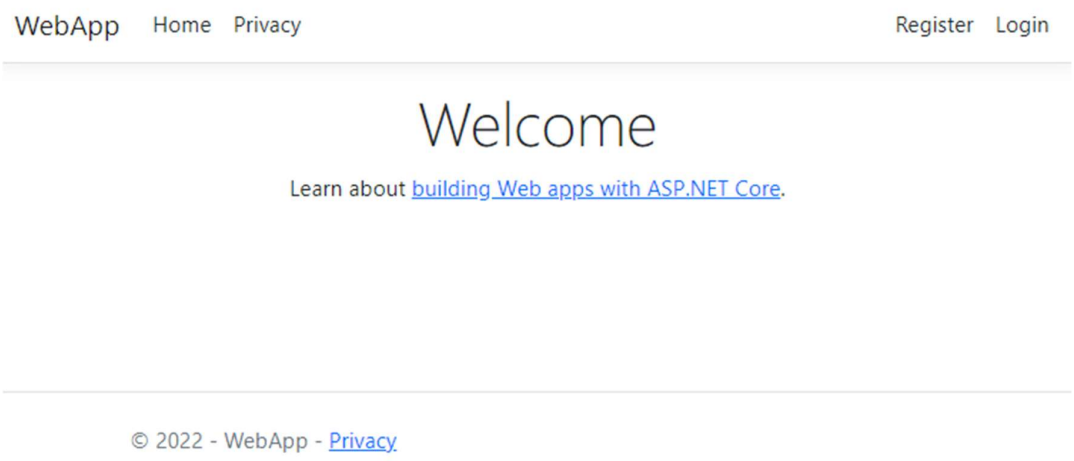
Loodud andmbebaasimudel on kujutatud joonisel 4.



Joonis 4. Rakenduse relatsiooniline andmebaasimudel.

5.2 Rakenduse kliendipoolne arendus

Vastloodud rakendust sisseehitatud veebiserveri abil käivitades avaneb kohe kohalikus hostis esialgne minimaalse kujundusega kasutajaliides pealehe ning sisselogimis- ja privaatsuseeskirjade lehega. Kirjeldatud kasutajaliidese vaade on toodud joonisel 5.



Joonis 5. Vastloodud rakenduse esialgne vaikimisi kasutajaliides.

5.2.1 Autentimine ja autoriseerimine

Individuaalseks kasutamiseks mõeldud rakenduse puhul on väga oluline isiklik autentimine, et teha kindlaks kasutaja identiteet. Individuaalne sisselogimise võimalus tekitatakse raamistiku poolt projekti loomisel tehtud valiku alusel. Vaadeldavas rakenduses kasutatakse autentimiseks kasutajanime ja parooli.

Kuigi osad rakenduse vaated on kõigile vaatamiseks avatud, saab tänu autentimisele piiritleda, millisteks tegevusteks peaks kasutaja sisse logima. Kasutaja õiguste kontrollimiseks toimub rakendusesisene autoriseerimine. Kuna rakendus toetab eri rolle (kasutaja, admin, nõustaja), siis on oluline, et serveri poolel oleks vaated rolli järgi eristatud. .NET raamistikus on võimalik kontrolleritele ja eraldi kontrolleri meetoditele lisada annotatsioonid, mis on toodud joonisel 6.

```
[Authorize]
[AllowAnonymous]
[Authorize("admin")]
```

Joonis 6. Rakenduse kasutajate rolliõiguste kontrollimise annotatsioonid.

Annotatsioonide alusel määratakse, milliste õigustega rakenduse kontrollerite eri meetoditele ligi pääseb. Rakenduse vaikimisi käitumine on õiguste puudumisel kuvada veateadet või suunata kasutaja sisselogimise lehele. Selleks et kasutajat ootamatustest säästa, tuleks õiguste kontroll lisada ka vaadetesse, nii et kasutajal ei oleks võimalik üldse kasutajaliideses tavalisi toiminguid tehes valesse kohta sattuda. Selleks on .NET raamistikus kasutajaõiguste kontrolli klass ClaimsPrincipal. Toodud koodinäites on määramata rollis võimalik kõigil klõpsata lingil Details, kuid ainult admini-rollis oleval kasutajal linkidel Edit ja Delete. Joonisel 7 on toodud koodinäide õiguste kontrollist rakenduste vaadetes.

```
<td>
    <a asp-action="Details" asp-route-id="@item.Id">Details</a>

    @if (User.IsInRole("admin"))
    {
        <a asp-action="Edit" asp-route-id="@item.Id">Edit</a>
        <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
    }
</td>
```

Joonis 7. Rakenduste vaadetes toimuva õiguste kontrolli koodinäide.

Kontrollerites ja vaadetes toimuva õiguste valideerimise abil saab määrata, milliseid vaateid kellele kuvatakse, hoides sel moel rakendusesisesed ligipääsud kontrolli all.

5.2.2 Kasutajaliidese disain

.NET veebirakenduste loomise raamistik ASP.NET on mugavasti koondanud rakenduse kontrollerid ja vaated kokku, nii on andmeobjektide edastamine vaadetele lihtne ja mugav.

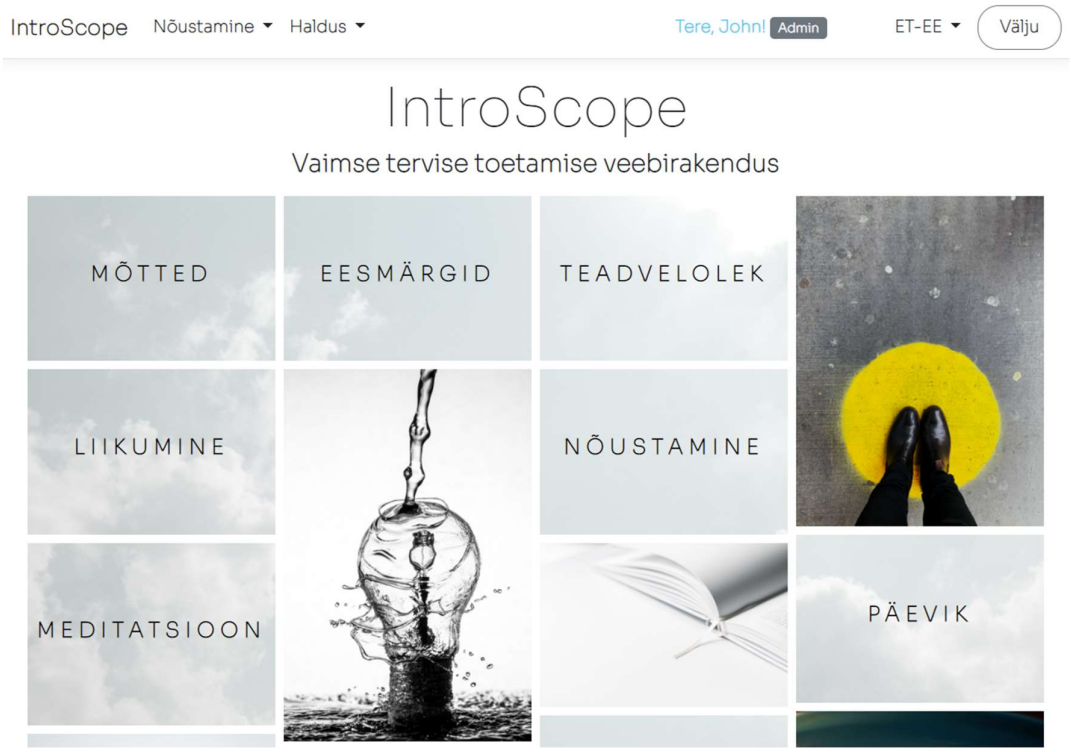
Lähtuvalt analüüsi käigus käsitletud kasutajakogemuse disaini põhimõtetest otsustas töö autor kasutada võimalikult minimalistlikku ja selget infostruktuuri, kasutajaliidese tõlkeid ning infomulle ja veateateid, et kasutaja saaks vajadusel lisainfot.

Samas on igal kasutajaliidesel ka esteetiline pool – rakenduse rõhk on toetada vaimset tervist ning kujundada n-ö pehmeid väärtusi. Ka rakenduse kujundus võiks värvi- ja pildikeelega seda visiooni kajastada. Seega tuli leida tasakaal põhiliste kasutajamugavust määravate faktorite ning esteetiliste kaalutluste vahel.

Kuna käibetõena teadaolevalt ütleb pilt rohkem kui tuhat sõna, siis oli üks võimalus kasutada kujunduses näiteks fotopankade vabaks kasutamiseks mõeldud kvaliteetseid fotosid. Kuid lihtne test rakenduse pealehel näitas, et suuremahulised pildid laevad aeglaselt, jättes lehest hangunud mulje. Seetõttu kasutatakse rakenduses fotosid vaid vinjettidena kohtades, kus nende laadimise viide minimaalselt häiriks ning rõhutakse pigem ühtse värvipaleti kasutamisele, et tagada järjepidevusega ühtlane stiil ja kasutaja jaoks meeldiv keskkond. Värvipalett loodi selleks mõeldud veebirakenduse coolors.co abil ning on toodud välja joonisel 8.

Joonis 8. Veebirakenduse IntroScope UI värvipalett.

Vastavalt disainiotsustele püüti hoida kontrasti tausta- ja tekstivärvi vahel. Samuti on välditud monoliitset kujundust, andes kasutajale aimu info jätkumisest lehe kerimisel. Kasutatud on piiratud koguses suuremahulisi fotosid ning värvipalett on minimalistlik, kuid aktsientvärvidega. Lähtuvalt kasutajakogemuse disaini analüüsist ja kujunduslikest otsustest loodi esialgne avalehe kujundus, mis on toodud joonisel 9.

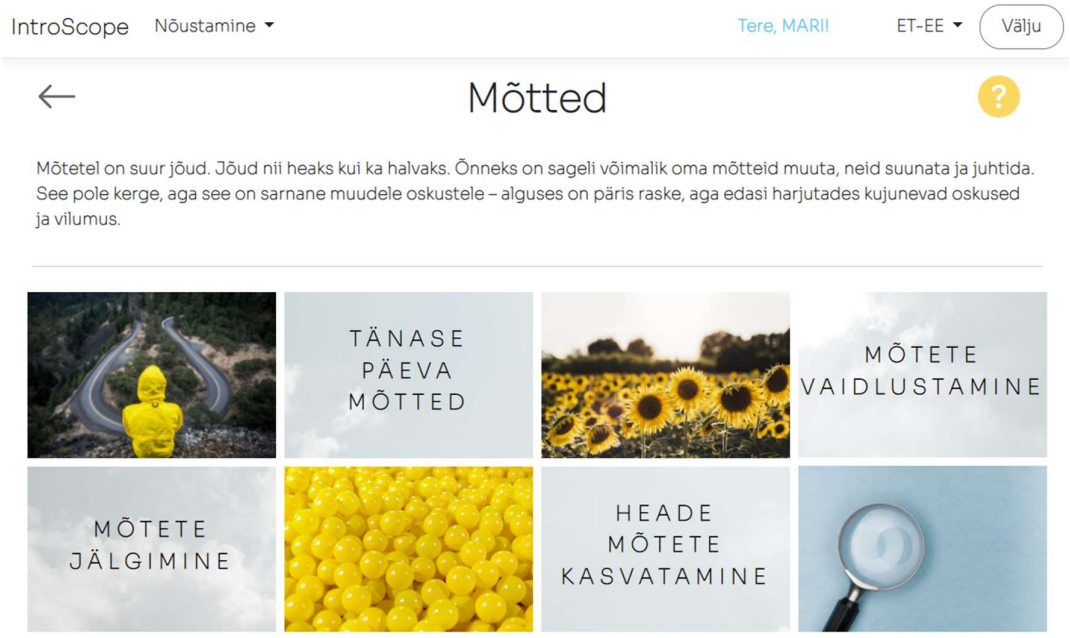


Joonis 9. Rakenduse IntroScope avaleht.

Samuti on pealehe menüü suurus reageerimisvõimeline, sõltuvalt ekraani suurusest, võimaldades kasutada ka mobiilivaadet. Näide kahandatud ekraaniga vaatest on joonisel 10.

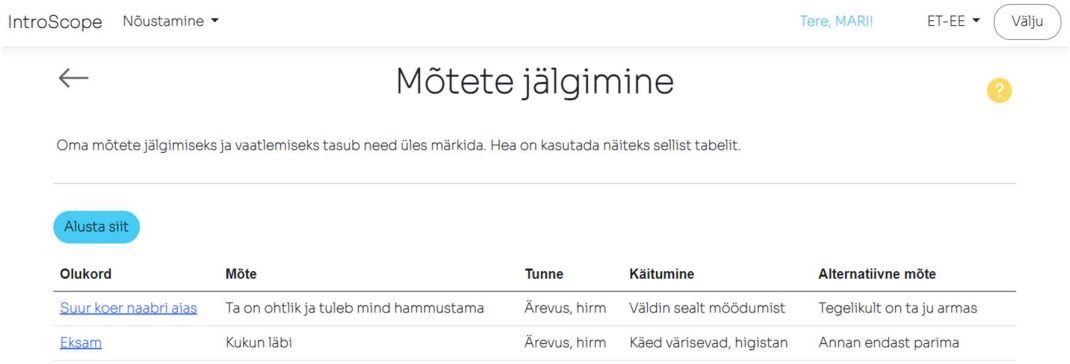
Joonis 10. Rakenduse kuva näide kahel eri suurusega väiksemal ekraanil.

Klõpsates pealehel mõnele kategooriatest, avaneb selle kategooria vormide vaade. Pealehelt edasi liikudes võimaldatakse kasutajal UI nooleikooni kaudu tagasi navigeerida. Lisaks on vormidel ka lisainfo nupp kollase küsimärgi ikoonina. Selle kohale liikudes avaneb lisainfo kastike. Kirjeldatud vaade kategooria „Mõtted“ vormide lehest on näidatud joonisel 11.



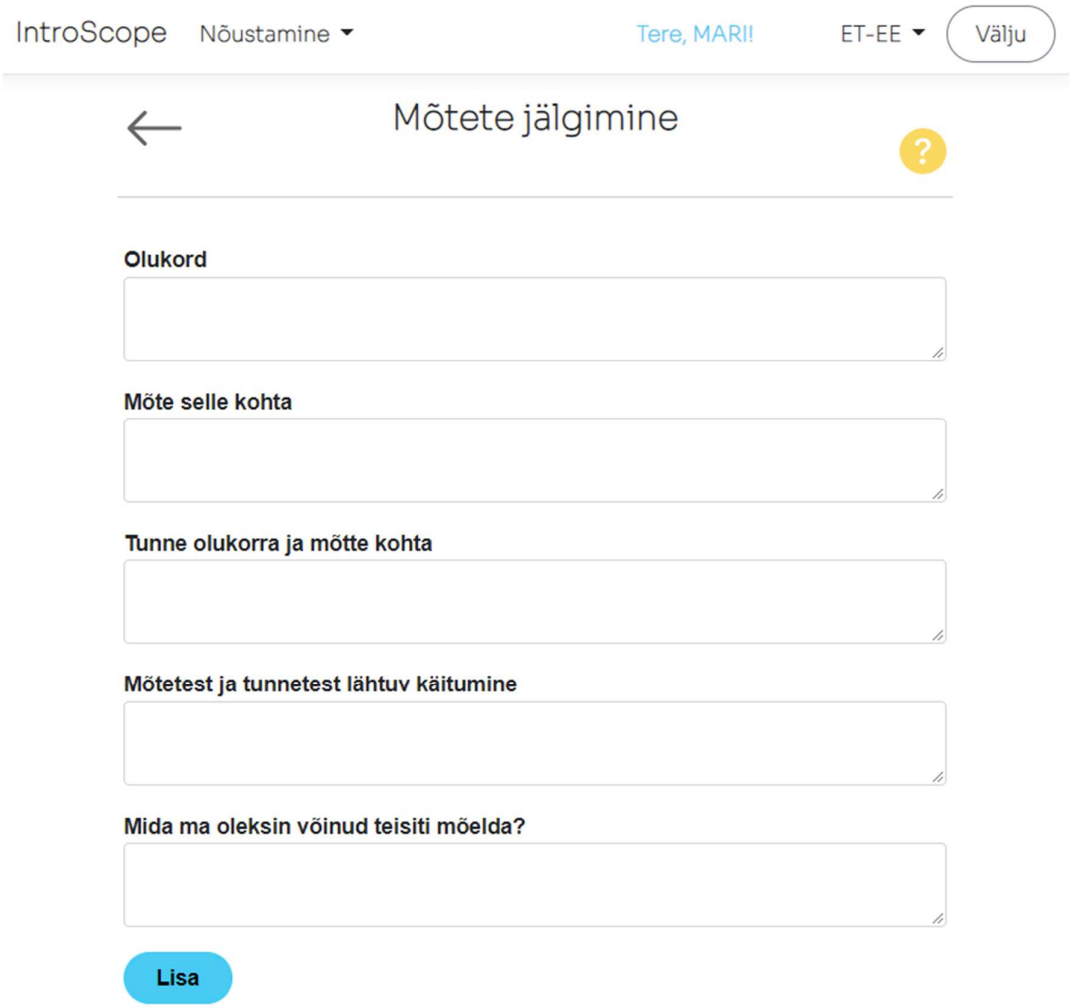
Joonis 11. Kategooria "Mõtted" lehe vaade.

Mõtete jälgimise vormile klõpsates avanevad kasutaja kõik varasemad sissekanded ning võimalus luua uusi. Vaade on kujutatud joonisel 12.



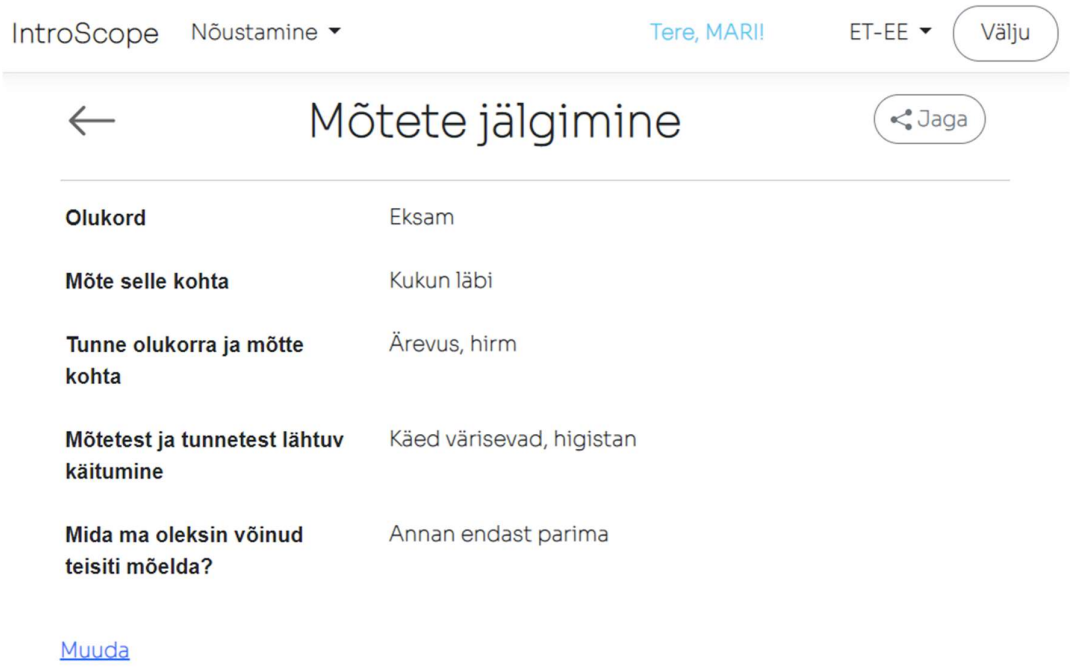
Joonis 12. Mõtete jälgimise kõigi sissekannete leht.

Klõpsates eelnevalt kirjeldatud vaate nupul „Alusta siit“, avaneb vormi täitmise vaade, mida on näha joonisel 13.



Joonis 13. Vormi täitmise vaade.

Salvestatud vormi saab avada detailvaates, kuhu on lisatud ka vormi jagamise nupp, nagu on näha joonisel 14.

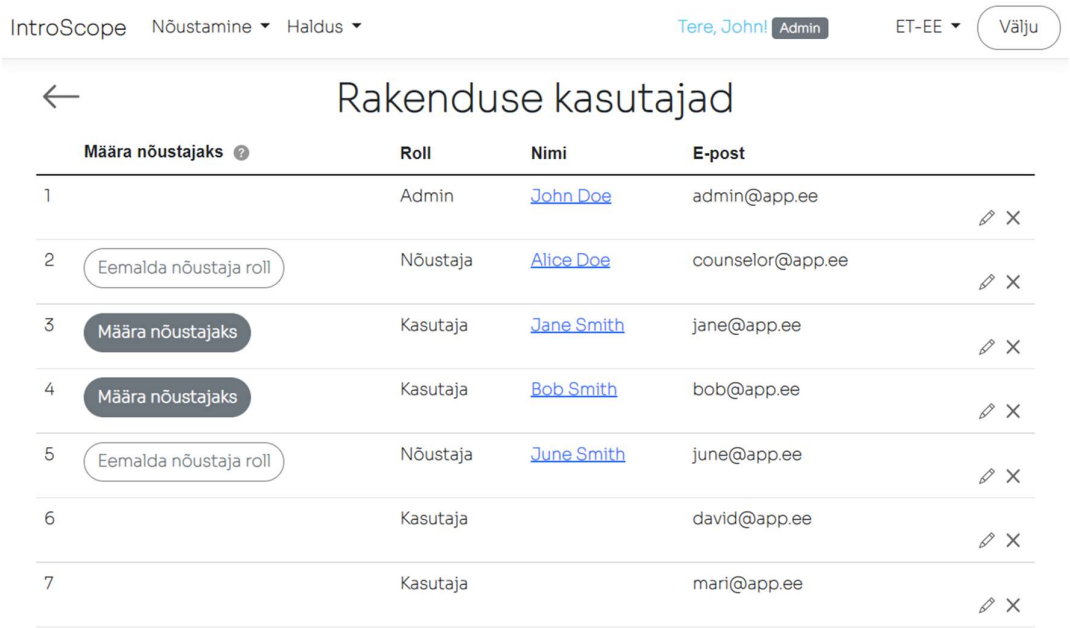


Joonis 14. Vormi detailvaade jagamise nupuga.

Rakenduse UI-s on läbivalt kasutatud infomulle kohtades, kus kasutaja võib vajada lisainfot mõne tegevuse kohta või siis täiendavat harivat teavet. Joonisel 15 on näide infomullist, mille abil kuvatakse täiendavat teavet.

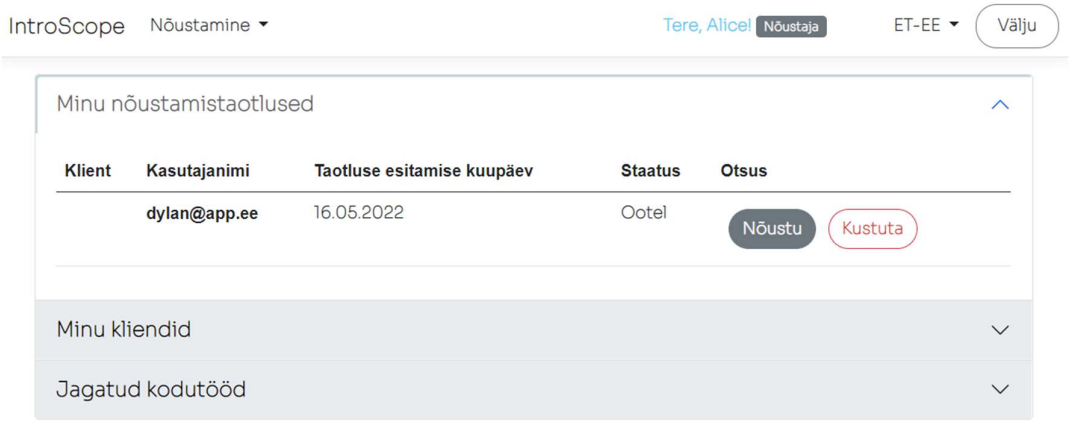
Joonis 15. Infomull täiendava teabega.

Rakenduse administraatoril on võimalik hallata kasutajaid ja rolle ülevaatliku tabeli abil, kus on infomullina täpsustus nõustajaks lisamise tingimuste kohta ning võimalus kasutaja nõustaja rolli määrata. Samamoodi on võimalik ka kasutajaid rakendusest eemaldada. Järgmisel joonisel 16 on toodud välja administraatorile kasutajate halduseks mõeldud vaade.



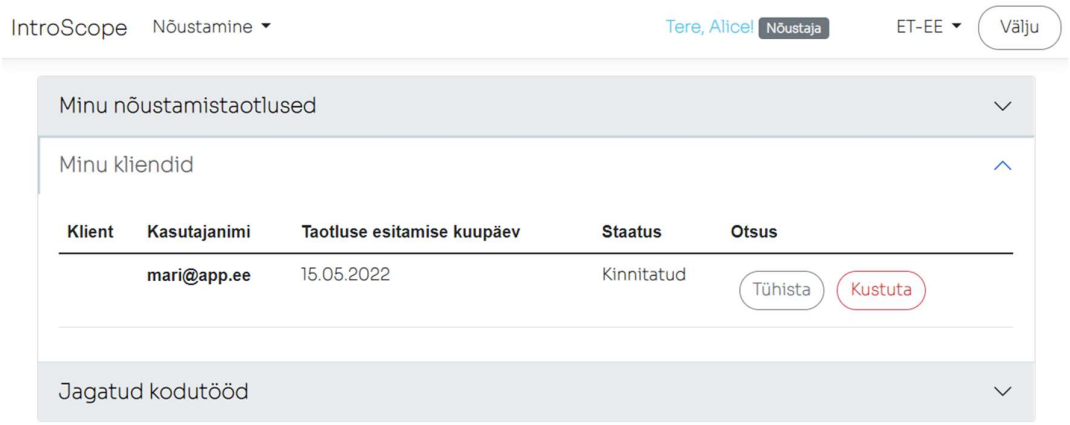
Joonis 16. Rakenduse administraatori vaade kasutajate haldamiseks.

Nõustaja rollis olevate kasutajate jaoks on loodud vaated nõustamisteenuse haldamiseks. Vaated on koondatud akordionielemendi sisse, nii et neid saab mugavalt teema kaupa avada ja sulgeda. Näide nõustajale saadetud nõustamistaotluste vaatest on joonisel 17.



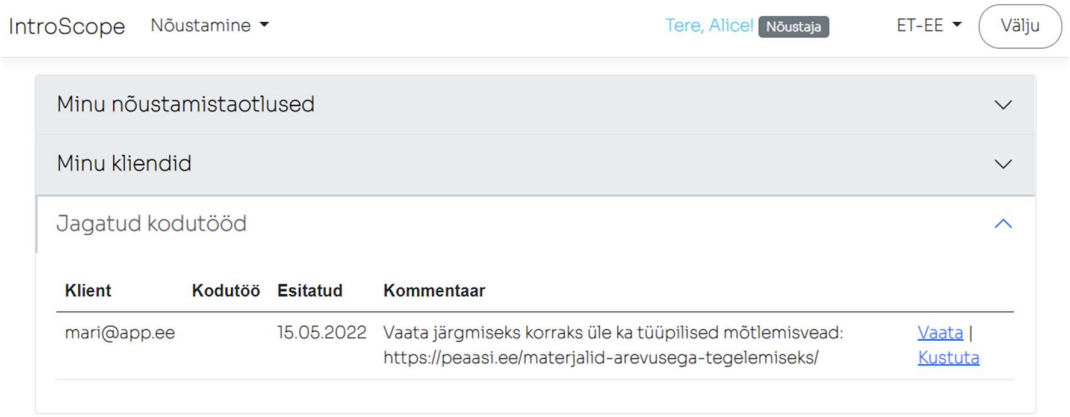
Joonis 17. Nõustaja nõustamistaotluste haldamise vaade.

Nõustamistaotluse kinnitamise korral lisatakse isik nõustaja kliendiks. Klientide haldamiseks on nõustajal eraldi vaade akordion-menüü sees. Selle vaate avamise korral on võimalik nõustamisel olevaid kliente hallata, nagu on näha joonisel 18.



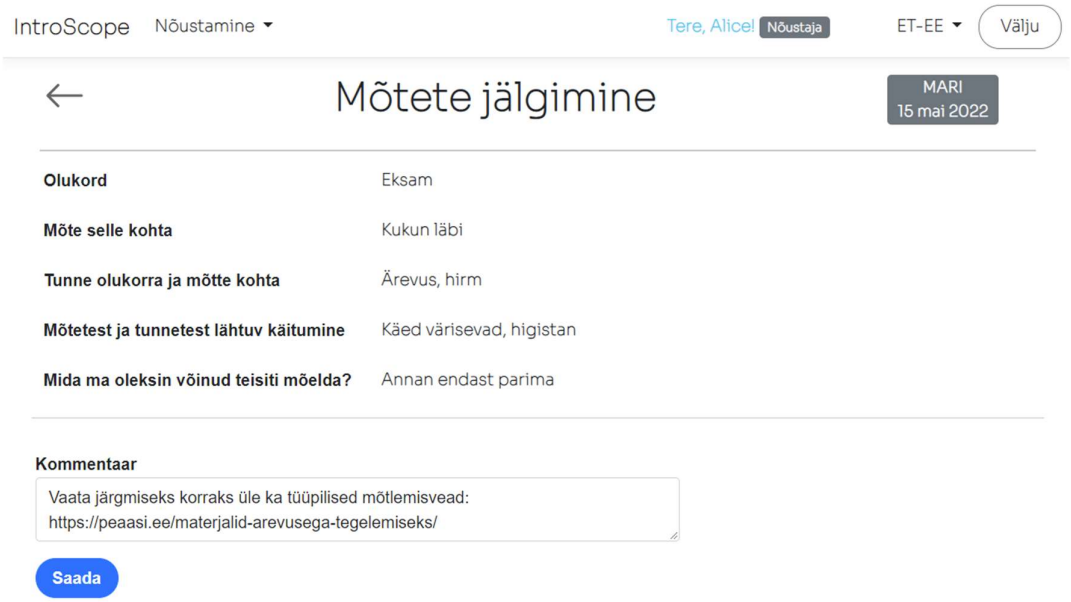
Joonis 18. Nõustaja klientide halduse vaade.

Samas akordion-menüüs on ka klientide jagatud kodutööde vaade (joonis 19), mille abil saab nõustaja ülevaate temaga jagatud eneseanalüüsi vormidest ning soovi korral neile lisada kommentaari.



Joonis 19. Nõustajaga jagatud kodutööde vaade.

Klõpsates kodutöö järel linki „Vaata“, avaneb nõustajale kodutöö detailvaade koos kommentaari lisamise võimalusega. Kirjeldatud vaade on toodud joonisel 20.



Joonis 20. Nõustajale avanev kodutöö vaade kommentaari lisamise võimalusega.

5.2.3 Tõlkimine

Selleks et rakenduse kasutatavust tõsta, võetakse kasutusele rakenduse UI tõlked. Nende haldamiseks loodi rakenduse projekt Resources, kuhu organiseeriti vastavalt vaadete asumisele failipuus ka nende tõlkelehed ehk resx-failid. Esialgu toetab rakendus inglise ja eesti keelt ning rakenduse UI kaudu saab keelt mugavasti vahetada.

5.2.4 Skaleerimine

Veebirakendust arendades oli esialgne eesmärk luua MVP ehk vähim elujõuline toode – olukord, kus rakendus täidab oma eesmärki vähimal määral, kuid piisavalt, et uuesti nõuded üle vaadata ja uus eesmärk seada.

Antud juhul on eesmärk luua võimalus täita UI kaudu ühte. Selleks vormiks valiti „Mõtete jälgimine“, mis pärineb portaali peaasi.ee pakutud trükisest „Ärevuse töövihik“ ning mille eesmärk on suurendada teadlikkust tüüpilistest mõttekäikudest, mis toimetulekut segada võivad.

Olles loonud andmeobjekti AlternativeThoughtForm koos selle juurde kuuluva kontrolleri, vaadete komplekti, repositooriumi ja teenusega, testiti UI kaudu vaadete läbimist ning andmete salvestumist korrektsel viisil andmebaasi. Õnnestunud esimese

etapi korral skaleeriti rakenduse funktsionaalsust, lisades eriteemalisi vorme samal moel juurde. Skaleerimise lihtsustamiseks loodi baasklassid BaseRepository ja BaseEntityService, mis koondasid vastavalt repositooriumite ja teenuste tavameetodid, mille iga andmemudeli individuaalne repositoorium ja teenus neilt pärisid.

Rakenduse mastaabi suurenedes lisandus ka tõlkeid ning kujundus, mida sai vastavalt algselt loodud MVP-le pisut täiendades taaskasutada.

5.2.5 Testimine

Rakenduse arenduse käigus ning lõppedes toimus pidev manuaaltestimine UI kaudu. Testiti läbi iga rolli põhitegevused rakenduses. Testandmete loomiseks tekitati veebirakenduse kausta klass AppDataHelper, mille meetodeid käivitades sai andmeid juurde lisada. Sel moel loodi eri rollides testkasutajaid ning ka vormide kategooriad, et hõlbustada tööd andmebaasi mudeli kujundamisel ja muudatuste järgset testandmete taastamist. Testiti läbi kõik kasutusvoo tegevused nii nõustaja, tavakasutaja kui ka administraatori vaadetes, et tagada andmete korrektne liikumine rakenduse UI-s ja andmebaasis.

5.2.6 Tagasiside kogumine

Loodud rakenduse kohta koguti tagasisidet kolmelt eri taustaga inimeselt: UI/UX disainile spetsialiseerunud tarkvara kvaliteedikontrolli spetsialistilt, psühholoogilt ning potentsiaalselt tavakasutajalt. Tagasisideks saadud kommentaarid on lisatud järgnevalt:

„Kohe näha, et tegu on projektiga, mille eesmärk on lahendada olulist probleemi. Lahendus on minimalistlik – välditud on liigset infomüra, seetõttu saavad ka need kasutajad, kes ei ole võib-olla kõige paremate tehniliste oskustega, veebirakenduses navigeerimisega hästi hakkama. Piltide kasutamine nuppudena on kasutajamugavuse poole pealt väga hea mõte. Rakendus sobib nii mobiili- kui ka tahvelvaate loomiseks, sest üha enam inimesi eelistab interneti kasutamiseks lauaarvuti asemel nutiseadmeid. Arendaja on valinud hea värvipaleti. On näha, et antud projektiga on nähtud vaeva ning tulemuseks on lihtne, kuid efektne rakendus, millele ei oleks probleemi leida kasutajaskonda isegi praeguses prototüübi faasis. Üliõpilane on saanud hakkama väga hea projektiga, mida võib probleemideta kasutada edasistes arendustes, mis võib kunagi ka viia reaalse kasutamiseni psühholoogide seas, nagu oli seda omal ajal ka eKool õpetajate seas. Tubli!“

Edgar Falilejev, tarkvara kvaliteedikontrolli spetsialist

„Loodud rakendus on psühholoogi töö lihtsustamiseks väga hea lahendus. Kodutööde haldamine on tehtud ülevaatlikuks ja mugavaks. Rakenduse kujundus on rahulik ja kasutajasõbralik. Rakenduse prototüüpi edasi arendades saaks pakkuda psühholoogidele mugavat töövahendit, mis teeks nõustamisteenuse töö interaktiivsemaks ja ajaliselt tõhusamaks. Samuti on see hea lahendus, kuidas kliendid saavad teraapiaväliselt ise enda vaimset tervist jälgida ja toetada. Ma loodan tõesti, et saan seda kunagi kasutada. Super töö!"

Hanna Reek, psühholoog

„Kindlasti kasutaks rakendust iseseisvalt, kui see oleks avalikult kättesaadav! Praegu teen väga palju märkmeid paberile, aga need kipuvad igale poole laiali minema. Kui oleksid erinevad vormid ja tavaline päeviku täitmise võimalus, siis oleksid kõik mu märkmed ühes kohas koos. See on ka hea, et rakendus on eesti keeles, muidu ei saaks võib-olla mõnedest sõnadest aru."

Piibe, õpilane (14 a)

6 Edasised arendused

Antud peatükis käsitletakse võimalusi veebirakenduse IntroScope prototüüpi käesoleva bakalaureusetöö skoobi väliselt edasi arendada ning avalikult kättesaadavaks teha.

Üheks oluliseks kaalutluseks rakenduse avalikult kättesaadavaks muutmisel on rahastusvõimaluste leidmine, mis võimaldaksid arendusprojektiga jätkata ning vajadusel tasulisi tehnoloogiaid kasutades rakendus avalikustada. Üheks võimaluseks oleks taotleda rahastust riiklike vaimse tervise edendamise projektide kaudu. Rakenduse rahastamisel ei oleks heaks lahenduseks reklaamipinna pakkumine, kuna rakenduse UX disainimisel on püütud vähendada UI infomüra, mis reklaamidega kaasneks.

Olles saanud projektile piisava rahastuse, oleks võimalik kaasata eri spetsialistid, et veebirakendus IntroScope täielikult välja töötada ning avalikult kättesaadavaks teha. Selleks oleks vaja töörühma, kuhu kuulub tooteomanik, kes määrab kindlaks rakenduse kasutajanõuded. Samuti kuuluksid sinna analüüsi, arenduse ja testimise rolle täitvad liikmed.

Töö tulemusena oleks võimalik luua vaimset tervist toetavad vormid koos nendega seotud kujundusega. Samuti on võimalik tõlkida rakendus näiteks vene keelde, et seda saaksid kasutada ka vene keelt kõnelevad isikud, näiteks sõjapõgenikud Ukrainast. Lisaks on võimalik arendada edasi administraatorile mõeldud kasutajaliidest, et rakenduse sisu mugavamalt hallata. Edaspidi oleks võimalik kaasata ka UI/UX spetsialist, et rakenduse disain ühtlustada ning kasutajakogemuse eri aspekte parandada. Rakendus peaks läbima ka põhjaliku turvatestimise, et tagada andmete terviklus, käideldavus ja konfidentsiaalsus.

7 Kokkuvõte

Käesoleva bakalaureusetöö raames loodi vaimset tervist toetava veebirakenduse IntroScope prototüüp. Rakenduse loomise ajendiks on vaimse tervise probleemide tõusutrend ühiskonnas ning sellega tegelevate spetsialistide ülekoormatus, mis on osalt tingitud halvasti korraldatud töövoost.

Olemasolevate lahenduste analüüsist selgus, et teadaolevalt puuduvad võimalused toetada vaimset tervist interaktiivsel moel erialaselt usaldusväärsete meetoditega ja eesti keeles nii iseseisvalt kui ka teraapia käigus ning seetõttu leiti, et selliste võimaluste pakkumiseks on vaja luua eraldi veebirakendus.

Rakenduse eesmärk on pakkuda võimalusi kasutajatel iseseisvalt vaimset tervist edendada ja psühholoogilist säilenõtkust suurendada ning soovi korral rakendust koos nõustajaga teraapiaprotsessi kodutööde haldamiseks kasutada. Eneseanalüüsi vormide loomisel võeti aluseks vaimse tervise spetsialistide välja töötatud materjalid.

Tehniliste võimaluste analüüsist tulenevalt valiti rakenduse arendamise raamistikuks .NET programmeerimiskeelega C# ning ASP.NET raamistik.

Tänu rakenduse pakutavatele võimalustele säästaksid psühholoogid ja kliendid igapäevatöös aega igalt kohtumiselt ligi kümme minutit, kuna nõustajal oleks võimalik eelnevalt kliendi kodutöödega tutvuda ning saadud teabele vastavalt järgmise kohtumise tegevusi planeerida. Lisaks võib klient olla kindlam, et paberkandjal tundlik info kuhugi ära ei kao ega valedesse kätesse ei satu. Need, kes iseseisvalt rakendust kasutavad, saavad võimaluse teadlikult oma vaimset tervist monitoorida, mõtteharjutusi läbi viia ning seeläbi elukvaliteeti tõsta.

Bakalaureusetöö raames käsitleti ka kasutajakogemuse disainiga seotud teemasid, et osata paremini näha rakenduse toimimist kasutajate vaatepunktist. Rakenduses võeti parema UX tagamise nimel kasutusele UI tõlked, selgitusi kuvavad infomullid ning

ekraanisuurusega kohanev kujundus. Läbimõeldud disainiotsustega püüti luua kasutajale turvaline keskkond, kus vaimset tervist toetada.

Loodud prototüüpi on võimalik edasi arendada toimivaks avalikult kättesaadavaks veebirakenduseks, mida võib näiteks riikliku rahastuse toel vaimse tervise keskustes kasutada. Rakenduse väljatöötamiseks tuleks luua vastav töörühm, mis hõlmaks vähemalt selliseid rolle, nagu tooteomanik, analüütik, arendaja ja testija.

Kasutatud kirjandus

[1] "The COVID-19 pandemic, financial inequality and mental health." *Mental Health Foundation*, 2020, [Võrgumaterjal] https://www.mentalhealth.org.uk/our-work/research/coronavirus-mental-health-pandemic/covid-19-inequality-briefing. [Kasutatud 13.03.2022]

[2] Abramson, Ashley. "Burnout and stress are everywhere." *American Psychological Association*, 2022, [Võrgumaterjal] https://www.apa.org/monitor/2022/01/special-burnout-stress. [Kasutatud 13.03.2022]

[3] "Maksa 100 eurot või oota mitu kuud: mida Eestis peale hakata, kui vaimne tervis teeb muret?" *Rahageenius*, 2019, [Võrgumaterjal] https://raha.geenius.ee/eksklusiiv/maksa-100-eurot-voi-oota-mitu-kuud-mida-eestis-peale-hakata-kui-vaimne-tervis-teeb-muret/. [Kasutatud 13.03.2022]

[4] Bruce, Timothy J., and Arthur E. Jongsma. *Adult Psychotherapy Homework Planner*. Wiley, 2021.

[5] "Mindfulness." *Mental Health Foundation*, 2021, [Võrgumaterjal] https://www.mentalhealth.org.uk/a-to-z/m/mindfulness. [Kasutatud 13.03.2022]

[6] The Mindfulness All-Party Parliamentary Group. *MINDFUL NATION UK. Report by the Mindfulness All-Party Parliamentary Group (MAPPG)*. The Mindfulness Initiative, 2015.

[7] Peaasjad MTÜ. "Noorte vaimse tervise portaal", [Võrgumaterjal] https://peaasi.ee/. [Kasutatud 13.03.2022]

[8] Bruce, Timothy J., Arthur E. Jongsma. *Adult Psychotherapy Homework Planner*. Wiley, 2021.

[9] World Health Organization. "Mental health." *WHO | World Health Organization*, [Võrgumaterjal] https://www.who.int/health-topics/mental-health#tab=tab_1. [Kasutatud 13.03.2022]

[10] Abramson, Ashley. "Burnout and stress are everywhere." *American Psychological Association*, 2022, [Võrgumaterjal] https://www.apa.org/monitor/2022/01/special-burnout-stress. [Kasutatud 13.03.2022]

[11] "Maksa 100 eurot või oota mitu kuud: mida Eestis peale hakata, kui vaimne tervis teeb muret?" *Rahageenius*, 2019, [Võrgumaterjal] https://raha.geenius.ee/eksklusiiv/maksa-100-eurot-voi-oota-mitu-kuud-mida-eestis-peale-hakata-kui-vaimne-tervis-teeb-muret/. [Kasutatud 13.03.2022]

[12] Bruce, Timothy J., Arthur E. Jongsma. *Adult Psychotherapy Homework Planner*. Wiley, 2021.

[13] Digital Documents, LLC. "The Most Common Paper Document Storage Issues And How To Solve Them." *Digital Documents, LLC*, 2016, [Võrgumaterjal] http://www.digitaldocumentsllc.com/the_most_commont_paper_document_storage_issues.htm. [Kasutatud 13.03.2022]

[14] Tervise Arengu Instituut. "Vaimne tervis." *Terviseinfo*, 2022, [Võrgumaterjal] https://www.terviseinfo.ee/et/valdkonnad/vaimne-tervis. [Kasutatud 20.03.2022]

[15] Careers In Psychology. "The Ten Worst Habits for Your Mental Health | CareersinPsychology.org." *Careers in Psychology*, [Võrgumaterjal] https://careersinpsychology.org/ten-worst-habits-mental-health/. [Kasutatud 13.03.2022]

[16] Blackwelder, Amanda. "Effect of Inadequate Sleep on Frequent Mental Distress." *Centers for Disease Control and Prevention*, 2021, [Võrgumaterjal] https://www.cdc.gov/pcd/issues/2021/20_0573.htm. [Kasutatud 13.03.2022]

[17] "Building your resilience." American Psychological Association, 2012, [Võrgumaterjal] https://www.apa.org/topics/resilience. [Kasutatud 09.05.2022]

[18] Barnett, Michaela, et al. "Good Habits, Bad Habits: A Conversation with Wendy Wood." *Behavioral Scientist*, 2019, [Võrgumaterjal] https://behavioralscientist.org/good-habits-bad-habits-a-conversation-with-wendy-wood/. [Kasutatud 14.03.2022]

[19] Chowdhury, Madhuleena Roy. "The Science & Psychology Of Goal-Setting 101." *PositivePsychology.com*, 2022, [Võrgumaterjal] https://positivepsychology.com/goal-setting-psychology/. [Kasutatud 20.03.2022]

[20] Gerry, Lisa M. "22 Mental Health Apps for Stress, Anxiety, and More." *Verywell Mind*, 2021, [Võrgumaterjal] https://www.verywellmind.com/best-mental-health-apps-4588479.

[21] "Web Content Accessibility Guidelines (WCAG) 2.1." W3C, 2018, [Võrgumaterjal] https://www.w3.org/TR/2018/REC-WCAG21-20180605/. [Kasutatud 01.05.2022]

[22] "Contrast Checker." WebAIM, [Võrgumaterjal] https://webaim.org/resources/contrastchecker/. [Kasutatud 01.05.2022]

[23] Richtel, Matt. "Making Web Sites More 'Usable' Is Former Sun Engineer's Goal." *The New York Times Web Archive*, 1998, [Võrgumaterjal] https://archive.nytimes.com/www.nytimes.com/library/tech/98/07/cyber/articles/13usability.html. [Kasutatud 20.04.2022].

[24] Nielsen, Jakob. “Top 10 Web-Design Mistakes of 2021 (Video).” *Nielsen Norman Group*, 2021, [Võrgumaterjal] https://www.nngroup.com/videos/top-10-web-design-mistakes/. [Kasutatud 20.04.2022].

[25] “Shneiderman's Eight Golden Rules of Interface Design – Capian.” *Capian*, [Võrgumaterjal] https://capian.co/shneiderman-eight-golden-rules-interface-design. [Kasutatud 20.04.2022].

[26] “Functional vs Non Functional Requirements.” *GeeksforGeeks*, 2020, [Võrgumaterjal] https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/. [Kasutatud 24.04.2022].

[27] B. Kohan, „Guide to Web Application Development,“ Comentum Corp, [Võrgumaterjal] https://www.comentum.com/guide-to-web-application-development.html.

[28] Microsoft. “ASP.NET | Open-source web framework for .NET.” *NET*, [Võrgumaterjal] https://dotnet.microsoft.com/en-us/apps/aspnet. [Kasutatud 24.03.2022].

[29] Tutorialspoint. “MVC Framework - Introduction.” *Tutorialspoint*, [Võrgumaterjal] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm. [Kasutatud 24.03.2022].

[30] “C# OOP (Object-Oriented Programming).” *W3Schools*, [Võrgumaterjal] https://www.w3schools.com/cs/cs_oop.php. [Kasutatud 7.04.2022].

[31] Hunt, Andrew, et al. *The Pragmatic Programmer*. Addison-Wesley, 2000. [Kasutatud 7.04.2022].

[32] “General Python FAQ — Python 3.10.4 documentation.” *Python Docs*, [Võrgumaterjal] https://docs.python.org/3/faq/general.html. [Kasutatud 7.04.2022].

[33] “Web Development Technologies – Orient Software.” *Orient Software*, [Võrgumaterjal] https://www.orientsoftware.com/technologies/web-technologies/. [Kasutatud 7.04.2022].

[34] *TypeScript: JavaScript With Syntax For Types.*, [Võrgumaterjal] https://www.typescriptlang.org/. [Kasutatud 7.04.2022].

[35] “MVC vs. Web API: Which ASP.NET technology should you use?” *Software Developer India*, 2018, [Võrgumaterjal] https://www.software-developer-india.com/mvc-vs-web-api-which-asp-net-technology-should-you-use/. [Kasutatud 10.04.2022].

[36] “Difference between ASP.NET MVC and ASP.NET Web API.” *Dot Net Tricks*, 2013, [Võrgumaterjal] https://www.dotnettricks.com/learn/webapi/difference-between-aspnet-mvc-and-aspnet-web-api. [Kasutatud 10.04.2022].

[37] “MVC vs. Web API: Which ASP.NET technology should you use?” *Software Developer India*, 10.04.2018, [Võrgumaterjal] https://www.software-developer-india.com/mvc-vs-web-api-which-asp-net-technology-should-you-use/. [Kasutatud 10.04.2022].

[38] “10 Best Source Code Management Tools For Version Control.” *Software Testing Help*, 2022, [Võrgumaterjal] https://www.softwaretestinghelp.com/best-source-code-management-tools/. [Kasutatud 10.04.2022].

[39] “Rider: The Cross-Platform .NET IDE from JetBrains.” *JetBrains*, [Võrgumaterjal] https://www.jetbrains.com/rider/. [Kasutatud 10.04.2022].

[40] “JetBrains Rider vs Visual Studio Code | What are the differences?” *StackShare*, [Võrgumaterjal] https://stackshare.io/stackups/jetbrains-rider-vs-visual-studio-code. [Kasutatud 10.04.2022].

[41] Martin, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Edited by Robert C. Martin, Prentice Hall, 2009.

[42] Dykstra, Tom, et al. “Kestrel web server implementation in ASP.NET Core.” *Microsoft Docs*, 2022, [Võrgumaterjal] https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-6.0. [Kasutatud 10.04.2022].

[43] “POCO Support - WCF.” *Microsoft Docs*, 2021, [Võrgumaterjal] https://docs.microsoft.com/et-ee/dotnet/framework/wcf/samples/poco-support. [Kasutatud 10.04.2022].

[44] “What is Entity Framework?” *Entity Framework Tutorial*, [Võrgumaterjal] https://www.entityframeworktutorial.net/what-is-entityframework.aspx. [Kasutatud 10.04.2022].

[45] Andres Käver. Building Distributed Systems Course material, TalTech 2021.

[46] Hunt, Andrew, et al. *The Pragmatic Programmer*. Addison-Wesley, 2000.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Kadri Jõgi,

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Vaimse tervise toetamise rakenduse IntroScope prototüüp“, mille juhendaja on Kristiina Hakk
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

16.05.22

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.