TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tamor Haabmets IABB 179791

IT TEENUSE JA TÖÖPROTSESSI VÄLJATÖÖTAMINE VILJANDI HAIGLA PERSONAALMEDITSIINI PROGRAMMI TOETAMISEKS

Bakalaureusetöö

Juhendajad: Dr Gunnar Piho
Martin Lall

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Tamor Haabmets

16.05.2020

Annotatsioon

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 5 peatükki, 4 joonist, 0 tabelit.

Abstract

Develpoment of IT service and working process for supporting Viljandi hospital's personal medicine program

The thesis is in Estonian and contains 25 pages of text, 5 chapters, 4 figures, 0 tables.

Lühendite ja mõistete sõnastik

IT	Infotehnoloogia
VH	Viljandi haigla
Task	Ülesanne. Rakenduse funktsionaalsus, mis tekitab ülesandeid küsimustike täitmiseks.
Target	Eesmärk. Rakenduse funktsionaalsus, mis näitab statistikat täidetud küsimustike kohta.
MVP	*Minimum viable product* ehk minimaalne töötav toode on esimene töötav versioon tootest, mida klient saab kasutada. [1]
POC	*Proof of concept*; idee esmane realiseerimine; tõestus, et kontseptsioon töötab
SLA	*Service level agreement* ehk Teenustasemelepe on kokkulepe teenuseosutaja ja kliendi vahel, milles määratletakse teenuse omadused: kvaliteet, kättesaadavus, vastutus jms. [2]
SLR	*Service level requirements* ehk „Teenustaseme nõuded on kliendi nõuete kirjeldus, mis tuleneb kliendi ja teenuseosutaja vahelisest koostööst“. [3]
ITIL	*Information Technology Infrastructure Library* ehk infotehnoloogia infrastruktuuride loetelu on infotehnoloogia haldamise tavade parimate praktikate kogu. [4]
DRP	*Disaster recovery plan* ehk suurõnnetusest taastumise kava „on dokumenteeritud ja struktureeritud plaan, mis kirjeldab, kuidas organisatsioon saab kiirelt jätkata oma tööga peale ettenägematut intsidenti.“ [5]
SCRUM	Scrum on iteratiivne agiilse tarkvara arendamise raamistik. [6]
API	*Application programming interface* ehk rakenduse programmeerimise liides „on tarkvara vahendaja, mis lubab kahel rakendusel üksteisega suhelda“. [7]
CRUD	*Create, Read, Uptade, Delete*; Loo, loe, uuenda, kustuta
Roadmap	*Roadmap* on tegevuse plaan, mida hakatakse kuna tegema.
CSV	*Comma Separated Values* on laialtlevinud andmeformaat
Stand-up	Lühikoosolek

Drag-and-drop	Tõmba ja kukuta
Textbox	Kirjakast, veebilehe osa, millesse saab teksti kirjutada
Front-end	Rakenduse osa, millega kasutaja otseselt kokku puutub
React	Front-end programmeerimise raamistik
Mahuhaldus	„Protsess, mis vastutab selle eest, et IT teenuste ja IT infrastruktuuri mahud oleksid piisavad teenustaseme sihtide saavutamiseks tulusal ja õigeaegsel viisil.“ [2]
Lansseerimine	Rakenduse avalikustamine veebikeskkonnas.
CMDB	*Configuration management database* ehk konfiguratsiooni halduse andmebaas, kus hoitakse infot projektis kasutusel olevat tarkvara ja riistvara kohta. [8]
CI/CD	*Continuous integration*, *Continuous delivery* ehk pidev integratsioon ja pidev üleandmine on parim praktika, kuidas koodimuudatusi ellu viia.
RACI	Vastutusmaatriks, milles kirjeldatakse ära projekti rollid ja vastutused. [9]
Devops	„On töökultuur, mis pakub paremat suhtlust erinevate IT-valdkondade vahel, mis loovad rakendusi ja teenuseid“. [10]
Proxy	On arvutivõrgu server, mis vahendab infovahetust kliendi ja serveri vahel. [11]
Repositoorium	Koodi hoidla

Sisukord

Jooniste loetelu

1 Sissejuhatus

Viljandi Haiglas (VH) täidavad meditsiinitöötajad (õed, arstid) küsimustikke patsientide kohta paberkujul.
Sellega kaasnevaid probleeme on mitmeid.

1. Andmete töötlemine on keerukas, suuremahulisemat analüütikat on peaaegu võimatu paberkujul andmetega teostada.
2. Täidetud küsimustike kontrollimine on ajakulukas meditsiini töötaja jaoks.
3. Küsimustike paberil täitmine on halb organiseering: paberid kaovad ära, käekirjaga on probleeme, küsimustikud on staatilised. Staatilisuse all on mõeldud, et mõndadele küsimustele peab vastama ainult siis kui on täidetud mõningad eeltingimused, kui aga eeltingimused täidetud ei ole, siis ei oleks vaja küsimust üldse kuvada, kuid paberil see võimalik ei ole.

Ma pakun haiglale arendusteenust, et teha digitaalne keskkond küsimustike täitmiseks. Eesmärgiks on disainida IT-teenus, mille osaks on veebipõhise liidesega rakendus, kus saab mugavalt täita digitaliseeritud dünaamilisi küsimustikke, et lahendada hetkel esinevaid probleeme küsimustike täitmisel Viljandi Haiglas. Dünaamiline küsimustik tähendab, et küsimustik võib täitmise käigus muutuda. Lisaks peaks rakendus suutma täidetud andmete põhjal automaatseid järeldusi teha ning nendest meditsiinitöötajat teavitama. Lahendus on mõeldud ainult andmete kogumiseks ja tööprotsessi edendamiseks, andmete analüüsimise funktsionaalsust sellel ei ole.

2 Metoodika

Projekti vältel oli mulle abiks mentor, kes on ka selle lõputöö juhendaja. Ta suunas ja abistas mind õigete metoodikate ja praktikate valikul ja implementeerimisel.

2.1 Agiilne lähenemine

2001 aastal ilmunud väljaandes „*Manifesto for Agile Software Development*“ [12] pandi kirja agiilse arenduse põhimõtted, mida ka selles projektis järgitakse. Agiilse arendusel on mitmeid eeliseid traditsioonilise arendamise ees, kuid põhiline põhjus, miks võtsime kasutusele agiilse arenduse on see, et agiilses arenduses tehakse esimesena see, mis võimaldab kliendil tuleviku otsuseid langetada. See kriteerium oli kliendi jaoks väga oluline ning koos sellega lahenevad ka muud probleemid, näiteks et tehakse midagi, mida kliendil tegelikult vaja ei ole.

Agiilse lähenemise valikuga kaasnesid mõned põhimõtted ja meetmed, mida me järgisime.

2.1.1 Kasutajalugude kirjapanek

Võtsime kasutusele kasutajalood ehk *User story-d*. See tähendab, et kui klient seletas meile oma nõudmisi, siis meie teenusepakkujatena panime kliendi vajadused väikeste lugudena kirja. Näitlikult tähendab see, et kui Juhan soovib teha protseduuri X, siis peab ta vajutama kohta A ja kirjutama kohta B. Säärased kasutajalood andsid kliendile hea ja kiire ülevaate sellest, kas me saime tema vajadustest õigesti aru. Lisaks viivad need kasutajalood ühisele lehele meid kliendiga ning seetõttu ennetataks tulevasi arusaamatusi.

2.1.2 Ärinõuete kaardistamine

Tundsime pidevalt huvi ärinõuete vastu. Kui klient soovis uut tehnilist lahendust, siis ei asunud me seda kohe arendama, vaid uurisime, mis on tema nõuded ja vajadused, et kindlaks teha, mida kliendil tegelikult vaja on. Süvenesime äriliselt igasse teemasse kuni olime piisavalt äri mõistnud, et anda nõu, kuidas probleemi kõige parem oleks

lahendada IT seisukohast. Keeruliseks tegi selle faktor, et osapooli oli projektis palju. Tuli arvestada kõikide soovide ja nõuetega, et lõpptulemus kõigi eesmärke täidaks.

2.1.3 Pidev töötava tarkvara tarne

Järgisime põhimõtet, et kliendi keskkonnas peab rakendus koguaeg töötama. Selleks oli tehtud eraldi arenduskeskkond, kus arendati uusi muudatusi ja alles siis kui need olid valmis, viidi uuendused kliendi keskkonnas sisse. Enne uuenduste tegemist, pidi alati ka vana versioon varundatud olema. Varundamine oli vajalik selleks, et kui midagi uuendamisel valesti läheb, siis oli võimalik vana töötava versiooni peale tagasi minna. Selline lähenemine tagas, et rakendus töötaks kliendi keskkonnas alati vigadeta.

2.1.4 SCRUM

Esialgsed kokkusaamised kliendiga olid harva, 1-2 korda kuus. Siiski projekti arenedes suurenesid töö mahud ja projekt läks oma loomult keerukamaks. Seega hakkasime praktiseerima SCRUM raamistikku. SCRUM on laialt levinud protsessi raamistik, millega jagatakse töid väiksemateks osadeks ning tehakse neid osasid väikeste ajavahemike kaupa, mida kutsutakse sprintideks. Sprint lõppeb tagasivaatusega tehtud tööle ning analüüsitakse, mis läks hästi ja mis halvasti [13]. Neid tagasivaatusi ehk *retrospektiive* tegime juhendajaga kahekesi, klienti kaasamata. Otsustasime kaks korda nädalas pidada kliendiga kuni poole tunniseid videokoosolekuid, mille sisuks oli, et mis oli tehtud viimase koosoleku möödumisest ning mida on järgmiseks vaja teha. Lisaks sai nende koosolekute käigus ka jooksvatele küsimustele vastused. Meie nädala esimeses osas olev koosolek oli rohkem rõhuga, et mida me tegema hakkame ja nädala lõpus olev koosolek oli rohkem kokkuvõtva iseloomuga. Seega kujunesid välja ühenädalased iteratsioonid ehk töötsüklid.

2.1.5 Ülesannete tükeldamine ja haldamine

Koos juhendajaga sõnastasime kõik kliendi vajadused ümber konkreetseks ülesanneteks, millel on selged piirid ning hinnatud töömaht. Töömahu hindamisel ei kasutanud me enamlevinuid praktikaid (näiteks planeerimispokkerit), sest selleks puudus praktiline vajadus ja lisaks võis ainult ühel liikmel aimu olla ülesanded sisust ja tehnilisest poolest. Tükeldatud ülesanded kirjutasime üles ja Jira tarkvara abil liigitasime neid kas ootel olevateks, tegemisel olevateks või tehtud ülesanneteks. Kui projekti käigus tekkis uusi ülesandeid, siis lisasime need kohe Jirasse ja pärast

otsustasime, kas antud ülesannet on üldse vaja täita või mitte. Säärane ülesannete haldamine andis hea ülevaate projektist ning aitas teha kliendile täpsemaid hinnapakkumisi.

2.1.6 Hinnapakkumised

Oma sisepoliitikast tulenevalt soovis klient enne igat etappi järgmise tehtava osa hinda teada, seega tegime iga etapi kohta neile hinnapakkumise. Pakkumises leppisime kokku, mis ülesanded järgneva etapi jooksul ära tehakse ning neist ülesannetest moodustus tundide arv, millest oma korda järeldus tehtava töö hind. Kuna kõiki kliendi nõudmisi ei olnud võimalik ja ka otstarbekas väga detailselt hinnata, siis paljuski oli pakkumises hinnavahemik, mis võiks antud ülesandele kuluda ja lõpphind kujuneb vastavalt kliendi soovidele. Antud hinnapakkumine ei olnud siiski kivisse raiutud kuni etapi lõpuni. See tähendab, et järgisime agiilse arenduse põhimõtteid ja kui kliendi prioriteedid muutusid või vaja oli hoopis midagi muud teha järgmisena, siis kohendasime hinnapakkumist vastavalt sellele.

2.2 Kasutatud meetodid ja praktikad

Projekti kolmandas ja neljandas etapis hakkasime kohandama projekti ITIL 4 standardite järgi. See tähendab, et koostasime SLA, SLR, mahuhalduse ja DRP. Need tagavad teenuse jätkusuutlikkuse ning hoiavad ära arusaamatusi probleemide esinemisel, sest vastutus ja tegutsemisjuhised on ära jagatud. Lisaks olid teenuse arendamisel peamiselt kasutusel *devops* võtted, arendust koodi kirjutamise mõttes oli vähe. See polnud algset planeeritud viis, vaid kujunes projekti jooksul selliselt välja, sest koodi kirjutamise vajadust niivõrd palju polnud.

2.3 Töövahendite valik

2.3.1 Toggl (Tööle kulunud aja mõõtmine)

Kuna töö oli tasustatud, oli tarvis tööle kulunud aega mõõta. Selleks võtsin kasutusele mugava aja mõõtmise rakenduse Toggl. Valisin selle seetõttu, et olin ülikoolis, Toggl’it juba kasutanud ning see töötas hästi minu jaoks.

2.3.2 Jira (Projekti haldamine)

Jira on Atlassiani toode, mis on mõeldud tehtavate ülesannete jälgimiseseks ja projekti haldamiseks [14]. Kuna mu juhendaja oli selle tarkvaraga kogenud, siis valisime projekti haldamise tööriistaks just Jira.

2.3.3 Confluence (Teabebaas)

Confluence on VH poolt võimaldatud teenus, kuhu laadisime üles kõik dokumendid, koosolekute protokollid ja igasuguse muu info, mida võiks tulevikus vaja minna. Sinna keskkonda olid lisatud kõik projektiga seotud liikmed, kellel kõigil on võimalus kasutada sinna laetud materjale.

2.3.4 Docker (Rakenduse jooksutamine)

Docker on tööriist, et paigaldada ja jooksutada rakendusi [15]. Kasutasime Dockerit, et üles seada küsimustike keskkonna teenus. Otsustasime Dockeri kasuks, sest see ei võta palju ressurssi rakenduse jooksutamiseks võrreldes operatsioonisüsteemil rakenduse jooksutamisega ning selle abil saab vähese vaevaga teenust ükskõik, mis tingimustel üles seada. Lisaks toetab Docker mikroteenuste arhitektuuri, mida võib tulevikus vaja minna [16].

3 Tulemused

3.1 Etappidesse jagamine (roadmap)

Töö tegemise käigus kujunesid välja eraldi projekti arenemise etapid.

3.1.1 Algus

Klient soovis, et ei hakataks nullist uut programmi arendama, vaid tuleks kasutusele võtta olemasolevaid lahendusi. Põhjus oli selles, et esiteks on see odavam ning teiseks see on vähem ajakulukas. Kuna küsimustike digitaliseerimine on tüüpiline probleem, siis internetis leidub palju valmis lahendusi selle probleemi adresseerimiseks. Kõik lahendused olid siiski pisut erinevad ning neist tuli välja valida kaks parimat. Valiku kriteeriumiteks olid näiteks, et lahendusel oleks juba oma rollide ja õiguste süsteem välja arendatud või et andmed ei tohtinud salvestuda lahendusepakkuja andmehoidlasse, vaid et andmed saaks haigla andmebaasi salvestada. Lisaks oli oluline, et küsimustikke oleks lihtne koostada mitte IT-inimesel ja et lahendus võimaldaks juurdearendusi teha.

3.1.2 Etapp 1 ja 2

Kui kaks parimat oli välja valitud, siis soovis klient nende mõlema kohta kontseptsiooni tõestust (POC). Nendest moodustuski kaks esimest etappi: POC 1 ja POC 2. Kui mõlemad ideetõestused olid valmis ning kliendile ette näidatud, siis valis klient kumma prototüübiga edasi minna ning arendada sellest MVP.

3.1.3 Etapp 3

MVP arendamine oli kolmas etapp projekti arengus ning selle alla kuulus kliendi keskkonnas testimine rakenduse lõppkasutajate poolt reaalsete patsientide peal, kuid sinna ei kuulunud veel haigla infosüsteemiga liidestamine. MVP oli kliendi jaoks ka kriitiline otsustuskoht, sest saadi esimene tagasiside lõppkasutajatelt, et kas tehtud rakendus ka tegelikult lahendab olemasolevaid probleeme ning kas projekt on potentsiaalselt tulus kliendile.

3.1.4 Etapp 4

Kui kolmas etapp oli läbitud, oli järgmiseks sammuks rakenduse kasutusele võtmine, ehk siis tootmiskeskkonna üles seadmine. See tähendab, et koostöös kliendiga tegime

ITIL-i standarditele vastava dokumentatsiooni, et kindlustada rakenduse jätkusuutlikkus ja püsivus. Lisaks hõlmas neljas etapp kogutavate andmete standardiseerimist, et valmistada andmeid ette edasiseks analüüsiks.

3.2 Prototüüpide valimine

3.2.1 Otsus kaks lahendust teha

Klient, kelleks põhiliselt oli VH IT osakonna juht, oli välja pakkunud, et projektile võiks teha kaks alternatiivset lahendust ning selle tegemise käigus aru saada, missugused täpsemad nõudmised tekivad tulevasele rakendusele, et see lahendaks probleeme, mis Viljandi haiglal tolle hetke seisuga olid. Kuna juhendajat veel esimeses etapis mul abiks ei olnud ja taolist varasemat kogemust mul IT-projektidega ei olnud, siis nõustusin kliendi ettepanekuga

3.2.2 Esimene valik

Esimeseks prototüübiks valisin lahenduse nimega Form IO. Valisin selle seetõttu, et neil oli väga selge ja lihtne küsimustike koostamise kasutajaliides ja andmeid sai salvestada eraldi andmebaasi, mitte Form IO andmebaasi. Lisaks oli sellel teenusel veel madalad tasud, korralik dokumentatsioon, sobivus mitmete keelte ja raamistikega ja mitmeid lisafunktsionaalsusi, mis oleksid potentsiaalselt kasulikud. Näiteks oli juba kasutajate ja rollide haldamine sinna sisse integreeritud või oli võimalik kiiresti ja lihtsalt automaatset emailide saatmist arendada. Võtsin ka arvesse, et olen veel vähese kogemusega IT-vallas ning seetõttu oli suurepärane, et Form IO kohta oli palju õpetavaid videosid ja muud abistavat infot, et igast programmeerimisalasest takistusest üle saada. [17]

3.2.3 Teine valik

Teiseks lahenduseks soovitas klient võtta Aafrikas toodetud vabavara nimega Medic Mobile, mida kasutatakse kohalikes haiglates infosüsteemina. Aafrika tarkvara ei ole Euroopas väga levinud, sest nende ajalooline kogemus on lühike, millest võib tuleneda, et tarkvara kvaliteet on kehv. Lisaks ei ole neil kehtestatud standardeid ja nõudeid, mis Euroopa Liidus kehtivad, näiteks turvalisuse ja konfidentsiaalsuse nõuded. Kuid kui natuke rohkem süveneda, siis on näha, et seda tarkvara tootev kommuun on väga suur, neil on Githubis (koodi hoidlas) üle 10000 *commiti* ja nad on seda ühte toodet

arendanud juba üle kaheksa aasta (esimesed *commitid* ulatuvad nii kaugele). Sellest kõigest saab järeldada, et see toote arendamisel on kasutatud heakskiidetud praktikaid ja et selle rakendusega on palju vaeva nähtud ning seda tulemusrikkalt kui seda rakendust kasutatakse ja arendatakse ka veel 8 aastat hiljem. [18]

Negatiivseteks poolteks selle lahenduse puhul oli see, et dokumentatsioon oli väga mahukas ja esmapilgul segaselt struktureeritud. See nõudis pikemat süvenemist, et rakenduse funktsionaalsustest aru saada ning võis aimata, et selle rakendusega kaasneb suur õpikõver. Lisaks oli rakendus arendatud Angulari raamistikus, mis oli mulle võõras ning nii suure projekti puhul oleks ka koodi õppimine väga suure õpikõveraga mulle. Sellest hoolimata otsustasime, seda rakendust lähemalt uurida, sest see võis potentsiaalselt sobilik lahendus olla VH-le.

3.3 Etapp 1 – Form IO PoC

3.3.1 Küsimustiku koostamine

Form IO-ga alustamine oli lihtne. Tegin Form IO kodulehel endale kasutaja ja üsna pea sain seal leheküljel küsimustikke koostama hakata. Küsimustike koostamine käis tõmba-ja-kukuta (*drag-and-drop*) meetodil. See tähendab, et ma justkui ehitasin küsimustikku klotsidest kokku. Näiteks kui ma tahtsin teha küsimust kuhu vastaja peab ise kirjutama teksti, siis tõmbasin kirjakasti (*textbox*) klotsi enda küsimustikku. Selline küsimustiku koostamine oli küllaltki intuiivne ning sain esimese prooviküsimustikuga kiirelt valmis.

3.3.2 Rakendusse implementeerimine

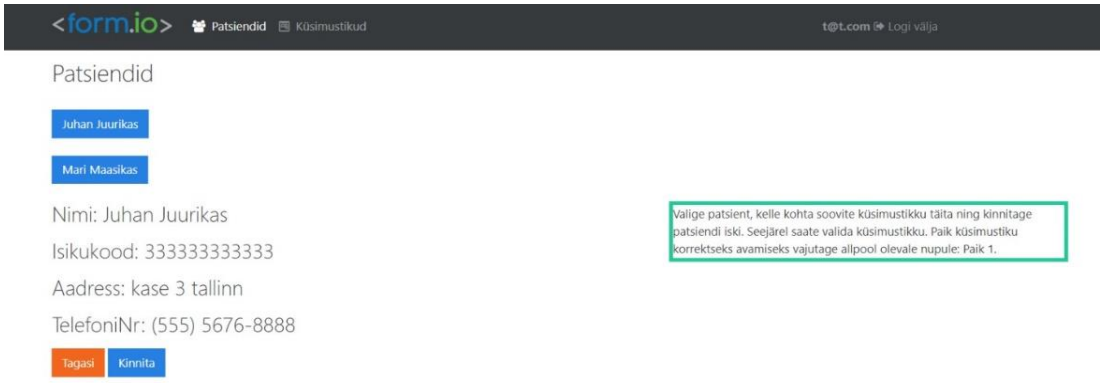
Seejärel tahtsin enda tehtud küsimustikku rakendusse implementeerida. Form IO toetas kõiki enamlevinud *front-end* programeerimise raamistikke ning ma valisin arendamiseks Reacti, sest mul oli sellega nii tööalaselt kui ka kooliga seoseskõige rohkem kogemust. Rakenduse arendamine läks väga lihtsalt, teenuspakkujal olid ette valmistatud käsuread, millega uus projekt genereerida. Projekti sees kuvasin ma eelnevalt koostatud küsimustikke ning olemasolevate õpetuste ja dokumentatsiooniga läks see suuremate takistusteta. Kui küsimustik ära täita, siis salvestusid andmed Form IO andmebaasi.

3.3.3 Kasutajad ja õigused

Form IO platvormil olid juba sisseehitatud kasutajate ja rollide süsteem, mida sai kodulehelt hallata. Lõin mõned testkasutajad ja andsin neile erinevad õigused küsimustikele ligipääsemiseks. Selle tagajärjel nägi kasutaja ainult neid küsimustikke, mis olid talle mõeldud nägemiseks. See oli kindlasti kliendile vajalik funktsionaalsus ja oli hea, et seda oli nõnda lihtne nende kodulehelt hallata.

3.3.4 Töövoo testimine API-ga

Klient kirjeldas, millisena ta näeb meditsiinitöötaja töövoogu. Kui töötaja läheb patsiendi juurde küsimustikku täitma, siis ta peab verifitseerima patsiendi isiku. Selleks otsib töötaja rakendusest patsiendi üles ning enne kui ta küsimustikku asub täitma, kuvatakse selle patsiendi andmed: tema isikukood, kodune aadress ja sünniaeg. Seejärel töötaja kontrollib töötaja patsiendi isiksust nende andmete abil ja asub peale seda küsimustikku täitma. Tegin kaks testpatsienti platvormi andmebaasi, et testida



Joonis 1. Form IO patsiendivaade

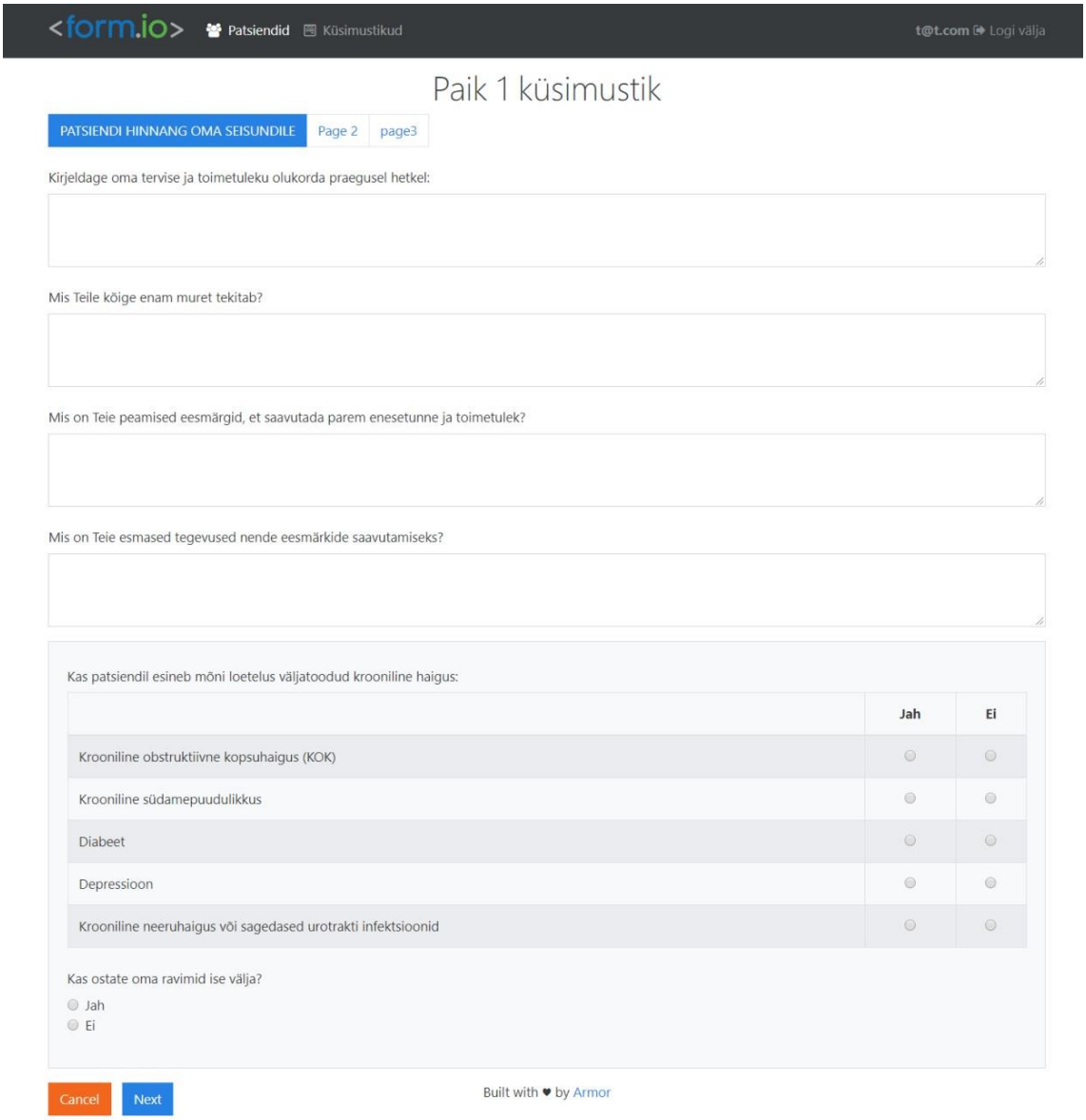
eelkirjeldatud töövoogu. Selleks, et patsientide info kätte saada, on teenusepakkujal välja töötatud API, mille vastu sai päringuid teha. Kuna dokumentatsioon oli hästi koostatud, siis ei võtnud kaua aega, et need päringud rakendusse implementeerida.

3.3.5 Rakenduse lansseerimine

Klient soovis, et tehtud rakendus oleks veebi kaudu ligipääsetav, et meditsiinitöötajad saaksid seda rakendust proovida ning esialgse tagasiside anda. Laadisin koodi Githubi üles ning tegin rakenduse avalikuks kasutades Netlify teenust.

3.3.6 Tulemus

Kliendile sai üle antud veebirakendus, millesse sai sisse logida, valida patsient, see patsient verifitseerida ja seejärel tema kohta küsimustik täita. Lisaks oli kliendi jaoks



Joonis 2. Form IO küsimustiku täitmise vaade

olemas ka juba rakenduse haldamise osa, mis oli lihtne ja arusaadav ka mitte IT-inimesele. Kliendi ärivajadus oli, et andmed ei salvestuks teenusepakkuja andmebaasi ja hetkel need siiski salvestusid, kuid tasulise paketiga oleks võimalik rakendus ümber seadistada kliendi andmebaasile.

3.4 Etapp 2 – Medic Mobile PoC

3.4.1 Rakenduse käivitamine

Medic Mobile näidisrakenduse tööle saamine oli keeruline ja ajakulukas. Nende dokumentatsioonis oli mitu erinevat moodust rakenduse tööle saamiseks, kuid iga moodusega tekkisid mingid vead, mida dokumentatsioonis polnud kirjeldatud ja ilma projekti sügavamalt tundmata oli võimatu lahendada. Seetõttu venis prototüübi valmimine.

Üks hetk lasi tootja rakendusest uue versiooni välja, mis muutis selle käimapaneku lihtsamaks. Rakendust oli võimalik üles seada, kas Dockeriga või siis arenduskeskkonnana Linuxi või MacOS operatsioonisüsteemis. Dockerit mul endal ei olnud võimalik arvutis installeerida ja ma pidin rakenduse üles seadma arenduskeskkonna. Kuna minu arvuti operatsioonisüsteemiks oli Windows, siis võtsin kasutusele Windowsi lisa nimega Windows sybsystem Ubuntu. Selle abil oli mul simuleeritud Linuxi operatsioonisüsteem arvutis, et keskkonda üles seada. Seega olid selles etapis peamiselt kasutusel *devops* arendusvõtted Selleks oli vaja eelnevalt luua lokaalne andmebaas. Rakendus kasutas andmebaasi mootoriks CouchDB-d. See on mitterelatsiooniline andmebaasi mootor nagu MongoDB, kuid ta on tasuta ja ilmselt seetõttu Medic Mobile-s kasutusele võetud. Kui andmebaas oli initsialiseeritud, siis sain rakenduse käivitada, kasutades Grunt tööriista. Lõpuks oli mul lokaalselt töötav rakendus ning sain katseda rakenduse funktsionaalsusi.

Kuigi mul ei olnud Dockeri võimekust, siis katsetasin Dockeri meetodit laenatud arvutil, kuid tööle see ei hakanud. Kui aga uus versioon välja tuli, siis sain ka Dockeri meetodiga rakenduse tööle.

3.4.2 Rakendusest

Rakendusel on olemas rakenduse põhifunktsionaalsuse pool ja rakenduse haldamise osa. Põhifunktsionaalsusi on viis.

1. Sõnumite saatmine rakenduse siseselt.

Rakenduse kasutajad saavad üksteisega kommunikeerida rakenduse siseselt. Selleks tuleb otsingusse kirjutada isiku nimi, kellele kirjutada soovitakse ning kui isik on leitud, tekib kastike, kuhu sõnum kirjutada. Kirjutatud sõnumid jõuavad kasutaja postkasti. Töötab sarnasel põhimõttel nagu Facebooki Messenger.

2. Hierarhiate loomine patsiendi, töötajate ja osakondade vahel.

Rakenduses on dünaamiline struktuuriüksuste ja inimeste vaheline hierarhia. Struktuuriüksused saavad kuuluda üksteise alla (näiteks mingi osakond kuulub mingi kliiniku alluvusse) ja igasse üksusse saab kuuluda inimene (kas patsient või töötaja). Struktuuriüksuseid ja inimesi saab kustutada, muuta ja ümber tõsta.

3. Küsimustike täitmine

Selleks, et küsimustikku täita, tuleb struktuuriüksuste hierarhiast üles leida inimene, kelle kohta soovitakse küsimustik täita. Seejärel on näha selle inimese andmeid, mille abil saab patsiendi verifitseerida (kui selleks on vajadus) ja peale seda valida küsimustik, mida täita soovitakse. Täidetud küsimustikke saab vaadata ja redigeerida.

4. *Taskide* kasutamine

Task ehk ülesanne on üldjuhul mingile kasutajale või struktuuriüksusele määratud küsimustik, mis vajab täitmist. See tekib mingitel teatud tingimustel. Näiteks kui patsient vastab küsimustikus teatud küsimusele teatud kindlal moel, siis tekib selle patsiendi kohta *task* täita järgmine küsimustik. Seda *taski* peab antud patsiendi eest vastutav inimene täitma. *Taskid* tuleb eeldefineerida arendaja poolt (see millal, missugune ja kellele *task* tekib).

5. Eesmärkide jälgimine (*target)*

Target ehk eesmärk on selleks, et jälgida jooksvalt täidetavaid küsimustikke. Näiteks tuli proovirakendusega kaasa selline küsimustik nagu „Sünnitus“, kus tuli vastata kuidas sünnitus läks. *Targetite* all oli siis jooksev statistika, mitu sünnitust on viimase 30 päeva jooksul toimunud, mitu neist olid õnnestunud ja mitu neist nurisünnitused. Sarnaselt *taskidele* tuleb *targetid* eeldefineerida arendaja poolt.

Joonis 3. *Targeti* näide

3.4.3 Rakenduse haldamine

Rakenduse haldamises on palju väikesi funktsionaalsusi ja mõned suuremad ja olulisemad. Väikesteks on näiteks keele ja kellaja formaadi muutmine, *targetite* sisse ja

välja lülitamine, piltide lisamine rakendusse, rakenduse konfiguratsioonidest varunduse tegemine. Tähtsamateks funktsionaalsusteks on kasutajate haldamine (CRUD operatsioonid), andmete eksportimine CSV formaadis ja rollide ja ligipääsude haldamine. Rolle saab luua ja kustuta ning igale rolli kohta on võimalik väga detailselt sätestada, mis funktsionaalsused talle rakenduses lubatud on. Lisaks on seal ka ülevaade rakenduses olemasolevatest küsimustike vormidest ja võimalus küsimustikke lisada.

3.4.4 Küsimustike koostamine

Medic Mobile dokumentatsioonist nähtub, et küsimustikke saab koostada mitut moodi. Küsimustiku koostamine ei tähenda automaatselt, et see rakenduses töötab, vaid see tuleb rakendusse lisada ja ka seda saab teha mitmel eri viisil. Rakendus toetab ODK XForm küsimustike formaati, ning sellises formaadis küsimustikke saab koostada vabalt valitud teenusepakkuja tööriista kasutades.

Kasutasin KoboCollect tööriista küsimustike koostamiseks. See oli funktsionaalsuselt väga sarnane Form IO *drag-and-drop* meetodile. Valmis küsimustiku eksportisin enda arvutisse, et seda hiljem rakendusse lisada. Seejärel tahtsin küsimustikud rakendusse laadida, rakenduse haldusvaate kaudu, kuid sellega tekkis takistusi. Rakendusele siiski ei sobinud see fail, mis KoBoCollect tööriist oli genereerinud, kuigi tootja ise oli seda tööriista soovitanud. Suutsin siiski veakoha tuvastada ning küsimustiku rakendusse üles laadida.

Teine võimalus oli küsimustiku koostamiseks oli kasutada meetodit, mida kasutab ka tootja ise. See meetod on küsimustike koostamine Excelis järgides XLSForms raamistikku. Kui Exceli fail on valmis, siis tuleb see küsimustik konverteerida rakendusele sobivasse (XForms) formaati. Selleks otstarbeks on tootja arendanud eraldi vahendi nimega Medic-conf. Kui Medic-confi abil on küsimustik konverteeritud saab selle sama tööriistaga küsimustikud rakendusse laadida, mis on teine võimalus küsimustike rakendusse laadimiseks. Esialgu ma seda meetodit ei katsetanud, sest lihtsam tundus esimese meetodiga alustada, kui hakata uut raamistikku selgeks õppima.

3.4.5 *Taskide* ja *targetite* koostamine

Taskide ja *targetite* koostamine on põhimõttelt sarnanane. See kätkeb endas JavaScript koodi kirjutamist dokumentatsioonis selgitatud põhimõtete alusel. Kui kood on valmis tuleb kasutada Medic-confi, et kompileerida üks suur rakenduste konfiguratsioonifail,

mis sisaldab *taske*, *targeteid* ja ka muud infot näiteks rollide, ligipääsude ja hierarhia kohta. Koostatud fail tuleb rakendusse laadida, kas siis nagu küsimustike puhulgi, Medic-conf abil või rakenduse haldamise vaatest. Proovisin seda kõike teha, kuid selle tulemusel lakkasid olemasolevad sisseehitatud *taskid* ja *targetid* töötamast.

3.4.6 Rakenduse konfigureerimine

Osa rakenduse haldamisest on mõistlik teha rakenduse haldamise vaatest, kuid paljusid asju saab teha ainult Medic-conf tööriistaga ning seetõttu oli selle vahendi vältimine võimatu. Medic-conf on käsurea liides, mille põhifunktsionaalsused jagunevad kahte blokki. Kõik funktsionaalsused on tootja poolt dokumenteeritud.

1. Küsimustikega seotud tegevused.

Nagu eelnevalt mainitud saab tööriistaga konverteerida küsimustikke Exceli formaadist rakendusele sobivaks formaadiks. Lisaks saab küsimustikke rakendusse lisada ja rakendusest kustutada (teistmoodi küsimustikke kustutada ei saa).

2. Konfiguratsiooni fail

Teiseks on võimalik Medic-confi abil võimalik rakenduse konfiguratsioonid kompileerida, rakendusse lisada ja olemasolevast konfiguratsioonist tagavara koopia teha.

3.4.7 Mitmekeelsus

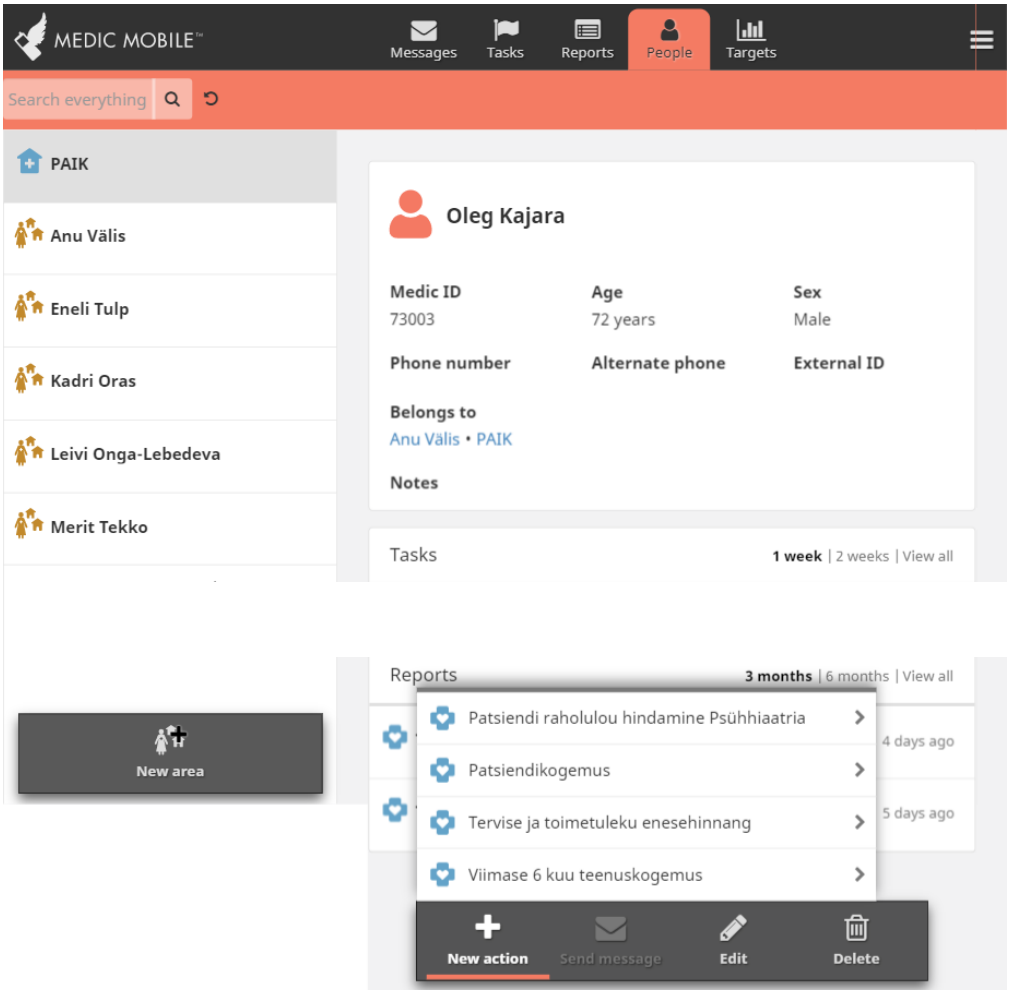
Rakenduses on olemas juba mitme keele funktsionaalsus. Uut keelt ja selle tõlkeid saab rakenduse haldamise süsteemist või konfiguratsioonifaili abil lisada ning seejärel saab kasutaja lihtsasti enda rakenduse keelt valida. Selle funktsionaalsuse alla kuuluvad ka küsimustikud, ka nende keelt on võimalik muuta.

3.4.8 Rakenduse avalikustamine

Ka selle prototüübi puhul soovis klient, et rakendus oleks veebi kaudu kättesaadav ning selleks kasutasin DigitalOceani platvormi. DigitalOcean pakkus kasutamiseks Linuxi virtaalmasinat, millesse oli juba eelinstalleeritud vajalikud Dockeri komponendid, seega seadsin Dockeri abiga veebilehe üles.

3.4.9 Tulemus

Medic Mobile on mahukas rakendus, mis lahendab mitmeid VH probleeme. Selle põhilised eelised on *taskide* funktsionaalsus ja juba töötav infosüsteemi infrastruktuur. Infrastruktuur on funktsionaalsused, mis pole otseselt küsimustike koostamise ja sisestamisega seotud, kuid vajalikud rakenduse normaalseks kasutamiseks (näiteks hierarhiate süsteem, rakenduse mitmekeelsus). Kuna funktsionaalsusi on väga palju, on ka vigu ja ebakõlasid rakenduses rohkelt ning paljud asjad ei pruugi esimese korraga õnnestuda. Kliendile sai demonstreerida küsimustike koostamise ja täitmise



Joonis 4. Medic Mobile patsiendivaade

funktsionaalsust. Lisaks ka sisseehitatud *taskide* ja *targetite* funktsionaalsust, kuid isetehtud *taske* või *targeteid* ei õnnestunud näidata.

3.5 Etapp 3 – Medic Mobile MVP

VH otsustas edasi minna teise prototüübiga nähes selles palju potentsiaali olemasolevate probleemide lahendamisel. Eriti kasulikuks pidas klient *taskide* funktsionaalsust.

3.5.1 Koostöö kliendiga

Koostöö kliendiga läks tihedamaks ning seetõttu hakkasime lühikoosolekuid (*stand-up)* korraldama. Hulk aega läks välja selgitamiseks, miks äripoolel mingit lahendust vaja on ja mida sellega saavutada üritatakse. Esialgu tekkis palju segadust ja möödarääkimisi. Näiteks alustasin tööga, mida klient polnud veel heaks kiitnud. Või siis ei saanud klient täpselt aru, millele oli tööaega kulunud ja mis oli selle töö tulemus. Arendusprotsess ei olnud kliendi jaoks piisavalt läbipaistev, kuid pannes rõhku projektihalduse tööriistade kasutamisele ja suhtlusele kliendiga, paranes olukord iga päevaga ning lõpuks oli nii teenuspakkuja kui klient rahul ja ühel arusaamal kõigist asjadest.

3.5.2 Ärivajaduste kaardistamine

Esialgu oli kliendile põhiprobleem, et *taskide* koostamine oleks läbi testitud. Peale seda soovis klient, et paigaldaksime töötava rakenduse ka kliendi taristus ning et haigla töötajad saaksid reaalsete andmetega rakendust testida. Selleks oli vaja koostada küsimustikud, mis klient ette andis ja need rakendusse sisestada. Lisaks oli vaja luua hierarhia, mis kataks kliendi ärivajadusi. Viimaks soovis klient dokumentatsiooni, kuidas rakendust nullist üles seada ja muu olulise info kohta rakendusest, mida ma olin projekti arendades teada saanud. Kirjutasin seda järgides põhimõtet, et kui mina selle projektiga edasi ei tegeleks, siis saaks keegi minu töö sujuvalt üle võtta. Need olid põhilised tükid, milles leppisime kliendiga kokku, mis olid vajalikud kolmandas etapis.

3.5.3 Arenduskeskkonna ülesseadmine

Kuna klient soovis esmajärjekorras, et *taskide* loomise funktsionaalsust saaks testida, oli vaja rakenduse praegusele veale lahendus leida. Ma ei osanud kuidagi sellele probleemile läheneda ning sel hetkel ma kutsusingi juhendaja appi. Tema soovitas rakendust Dockeri teenusena üles seada, sest siis on vigade leidmine lihtsam, efektiivsem ja mugavam. Kuna tulevikus oli vaja ka rakendus Dockeri teenusena kliendi keskkonnas üles seada, siis oli see niikuinii vajalik töö. Selleks laenas juhendaja enda virtuaalmasinat ning installeeris sinna Ubuntu ja Dockeri, et kõik eeldused oleksid täidetud rakenduse ülesseadmiseks.

3.5.4 Dockeris käivitamine

Dockeriga käivitamine arenduskeskkonnas ei läinud probleemide vabalt. Rakendus ei läinud tööle, vaid kuvas mitte midagi ütlevat veateadet. Kirjutasin tootjatele selle probleemiga seoses, kuid nad ei osanud lahendust pakkuda, sest nad ei osanud viga reprodutseerida. Mõne aja möödudes leidis juhendaja lahenduse ning saime rakenduse tööle. Peale seda avastasin ka miks *taskid* korralikult ei töötanud. Medic-conf oli rakenduse seaded valesti kokku kompileerinud. Dockeriga installeeritud rakenduse seadetel olid need vead parandatud ning töötava näidise abiga sain tööle *taskide* ja *targetite* koostamise funktsionaalsuse.

Selgus veel üks probleem. Kui rakendusega seotud internetidomeenile ei ole lisatud verifitseeritud sertifikaati, siis ei saa ükski kasutaja peale administraatori sisse logida ehk siis rakendus ei tööta korrektselt. Selle probleemiga seoses kirjutasin uuesti tootjatele ning nad pakkusid välja ajutise lahenduse, mille suutsin kiirelt implementeerida.

3.5.5 Rakenduse lansseerimine kliendi keskkonnas

Dockeri eelis on see, et igas keskkonnas saab rakenduse lihtsasti üles seada [19] ning seda ma ka tegin. Klient tegi sarnaselt juhendajaga mulle virtuaalmasina, milles sain kiiresti rakenduse käima panna. Lisaks palusin kliendil luua domeen ja sertifikaat sellele domeenile. Sellest keskkonnast sai testkeskkond, millele pääses klient ligi ning milles viisin muudatusi läbi alles siis kui olin need muudatused arenduskeskkonnas läbi proovinud. Viimaks tekitasime esmase konfiguratsioonihalduse (CMDB), et oleks rakenduse lansseerimine oleks taaskorratav samade parameetritega.

3.5.6 Turvalisus

Kuna rakendust oli plaanis testida päris andmetega, tuli mõelda ka andmete turvalisusele. Aafrikas ei ole IT-rakendustel nii kõrgeid turvanõudeid nagu Euroopas ning seda oli ka rakendusest näha. Näiteks hoiustati kasutajate salasõnasid tavatekstina mitte krüpteerituna, mis on halb praktika küberturbe vaatepunktist [20]. Bruce Schneider on öelnud „Kui sa arvad, et tehnoloogia saab lahendada sinu turvalisuse probleemid, siis ei mõista sa probleeme ja ei mõista ka tehnoloogiat“ [21]. Ka siin projektis on paljudele turvalisuse probleemidele lähenetud protseduuriliselt. Esmase turvalisuse tagamiseks:

- Said selgelt dokumenteeritud vastutuse piirid ja rollid. Näiteks annab IT tellimuse alusel infrastruktuuri (virtuaalmasina, veebiaadressi) teenuseosutajale.

- Välja sai töötatud protseduurid ja juhised, kuidas IT avab teenuse äripoolele. See tähendab, et on ainult üks inimene, kes saab luua uusi kasutajaid ja anda neile õigusi.

- Lisasin keskkonnale tulemüüri piirangud, et ainult lubatud teenused ja pordid oleksid võrgus avatud, seega on rakendus kättesaadav ainult VH sisevõrgust.

- Tellisin veebisertifikaadi teenuse domeenile.

- Teenus on avalikustatud läbi VH *proxy*, mida monitooritakse VH poolt. *Proxy* jälgib liiklust veebi ja rakenduse vahel ning kui märkab kahtlast tegevust (näiteks on üritatud 100 korda järjest sisse logida), siis annab märku VH töötajatele sellest.

3.5.7 Tulemus

Klient sai täiendatud turvameetmetega testkeskkonna, millega said rakenduse lõppkasutajad rakendust reaalsete andmetega testida. Paranes ka projekti selgus ja arusaam nii kliendi kui teenusepakkuja jaoks, millesse panustas pidev suhtlus, sobivate tööriistade kasutamine ja dokumentatsiooni kirjutamine kogu tehtud töö kohta.

3.6 Etapp 4 – Rakendusest teenus

3.6.1 Andmete standardiseerimine

Selleks, et töötajad rakendust kasutama saaksid hakata oli vaja enne standardiseerida tekitatavaid andmeid selliselt, et analüütik saaks neid andmeid pärast efektiivselt kasutada. See protsess on hetkel pooleli. Jõutud on sinnani, et milline oleks kõige õigem formaat andmete jaoks.

3.6.2 ITIL standardi järgimine

Tootmiskeskkond tähendab seda, et rakendus peab töötama teatud ajal teatud tingimustel. See omakorda tähendab, et erinevate probleemide tekkimisel ja muudatuste läbiviimisel on kindel tegevusplaan, sest muidu ei saa tagada rakenduse püsivust ja

pidevust. Selleks koostasime dokumendid, mis selgesõnaliselt panid paika nõuded rakendusele ja tegevusplaanid erinevate sündmuste korral. Lähtusime ITIL standarditest ja koostasime SLA, SLR, DRP ja mahuhalduse plaani. Alustasime isikute täpse tuvastamisega ning nende rollide kaardistamisega. Selleks kasutasime ITIL-i standardi RACI maatriksit [22]. See loob aluse, et teenus oleks läbipaistvam ja auditeeritavam. Ka need protsessid on hetkel pooleli, näiteks DRP dokumendid on valmis, kuid kliendi poolt kinnitamata ja testimata.

4 Analüüs

4.1 Etapp 1

4.1.1 Hea, kuid liiga lihtne

Minu kui teenusepakkuja vaatepunktist õnnestus esimene prototüüp väga hästi. Kõik oli lihtne, arusaadav ja minimalistlik. Samas oli see lihtsus samal ajal nii nõrkus kui ka tugevus, sest rakendus vajas väga palju täiendamist. Form IO ei olnudki mõeldud eraldiseisvaks rakenduseks, vaid koodi lisana, mis teeb juba mingid osad rakendusest ära. Kuna kliendi jaoks oli ajaline ressurss määrav, siis oli mõistetav, et esimene prototüüp polnud kliendi jaoks nii atraktiivne kui teine.

4.1.2 Peidetud väärtus

Esimesel etapil ei olnud käegakatsutavat väärtust, tehtud tööd ei võetud isegi vähesel määral kasutusse ning ei saadud otsest õppetundi või vajalikku infot projekti kohta. Siiski polnud selline projekti areng kasutu, sest esimese etapi jooksul arenes usaldusväärne suhe kliendiga ning lõime üheskoos esimesi metoodikaid koostöö tegemiseks, näiteks *User Storyde*, ja Confluence kasutamine. Selline väikeselt alustamine sobis ka minule hästi, sest mul puudus varasem kogemus paljudes asjades ning sain protsesse ja metoodikaid järk-järgult kasutusele võtta, mul oli aega nendega kohaneda. Võttes kõiki neid aspekte arvesse oli esimene etapp omal kohal ja prototüübile kulutatud ressurss ei läinud raisku, vaid tasus ennast ära.

4.2 Etapp 2

4.2.1 Rakenduse saab kiirelt kasutusele võtta

Medic Mobile oli edukas, sest see täitis lõppeesmärki paremini, ehk siis lahendas mitmeid kliendi probleeme. See oli karbitoode, mille saaks teoreetiliselt kohe kasutusele võtta ilma, et mingeid lisaarendusi oleks vaja teha. Kuna kliendi üks põhimuresid oli ,et nad tahavad võimalikult kiiresti infot koguma hakata, siis paistis selline karbitoode ideaalse lahendusena haiglale.

4.2.2 Haiglale disainitud

Tavaliselt on standardiseeritud lahenduste miinuseks see, et nende funktsionaalsused on liiga üldised ning ei lahenda olemasolevaid probleeme piisavalt efektiivselt [23]. Medic Mobile puhul oli positiivseks küljeks see, et rakendus on just haiglatele kasutamiseks disainitud. See tähendab, et rakendus arvestas haigla iseärasustega. Näiteks hierarhia osas kattus see täielikult VH olemasoleva töötajate ja patsientide hierarhiatega. Üheks haiglale sobilikuks funktsionaalsuseks olid ka *taskid*.

4.2.3 *Taskid*

Taskid aitasid töötajatel tööprotsesse parendada, sest paljude küsimustike puhul on vajalik järeltegevusi teha ning *taskid* aitavad neid tegevusi automaatselt delegeerida. Seetõttu toimuvad delegeerimised kiiremini ja mistõttu info patsientide kohta kogutakse kiiremini ja ka raviga saab kiiremini alustada, mille tagajärjel ravikvaliteet paraneb. Lisaks väheneb meditsiinitöötajate töökoormus, sest nad ei pea ise edasisuunamisi tegema. Tüüpiliselt on küsimustike vastustest vaja välja arvutada skoor, mille põhjal otsustatakse, kas patsient vajab lähemat uurimist või mitte. Töötaja ei pea ennast vaevama skoori arvutamisega, vaid neile tuleb ainult teade, et näiteks patsient vajab ekstra tähelepanu ja lisauuringuid. Viimaks väheneb vigade arv, mis on tingitud inimlikust eksimusest, sest rakendusse sisse kirjutatud algoritm ei eksi edasi suunamisel.

4.2.4 Töö areng

Töö protsesside ja arengu vaatepunktist oli rakenduse areng kohmakas ja aeglane. Tekkis palju takistusi, mille lahendamine ei olnud efektiivne ja kulutas palju ressursse. Näiteks oleks tulnud juhendaja juba varem appi võtta, sest aeg, mis ma kulutasin rakenduse uurimisele ei andnud tulemust. Olin ühe aspekti küljes kinni, kuid vajasin hoopis teistsugust lähenemist, mille juhendaja tõi. Arvan, et õigel ajal juhendaja kaasamisega oleks projekti areng 1 kuu võrra kiirenenud (praeguseks on projekt 8 kuud kestnud).

4.2.5 Kehv dokumentatsioon

Projekti areng oleks võinud minna sujuvamalt kui rakenduses ja dokumentatsioonis poleks nii palju vigu olnud. Seda rakendust iseloomustabki võimalus asju mitut moodi teha, kuid ainult üks moodus tegelikult töötab. See on tugevalt negatiivne külg, sest

esiteks võtab rakenduse konfigureerimine ja arendamine palju rohkem aega, kuid kiire projekti areng oli kliendi jaoks üks tähtsamaid faktoreid. Teiseks on raskem tööd üle anda uuele arendajale, sest lisaks originaalsele tootja dokumentatsioonile tuleb teha eraldi nö vigade paranduse dokumentatsioon.

4.2.6 Küsimustike koostamise keerukus

Üks lahenduse valiku kriteeriume oli, et küsimustikke saaksid koostada inimesed, kellel pole tugevat IT tausta. VH soov oli, et nad ei peaks iga uue küsimustiku tegemisega teenusepakkuja poole pöörduma. Medic Mobile seda nõudmist ei rahuldanud, sest küsimustike koostamine eeldas programmeerimise raamistiku tundmaõppimist. Positiivse poolena, suure keerukusega kaasnes see, et küsimustike konfigureerimise võimalused olid väga mitmekülgsed. Seega efektiivse seadistamise abil oli võimalik luua unikaalseid lahendusi, mis tavapäraselt poleks võimalik või nõuaks suurt lisaarendust.

4.2.7 Töö mahu ja maksumuse hindamine

Enne igat etappi soovis klient tehtava töö hinda teada ning selleks tuli ära hinnata tehtava töö maht. Töö mahu hindamine oli teises etapis väga robustne. Kuna ma ei teadnud isegi täpselt, mida Medic Mobile endast täpselt kujutab. Hindasin üsnagi empiiriliselt ning mitte millelegi tuginedes tehtava töö mahtu. Tehtav töö ei olnud kliendiga väga selgelt kokku lepitud, sest ka klient soovis rakendusest rohkem teada saada ja seetõttu klient ei teadnud, mida ta täpselt soovida saaks. Selline umbropsu hindamine lõppes kehvasti, sest tegelik töö maht oli poole suurem kui hinnatud töö. Õnneks suutsime kliendiga jõuda kompromissini kulutatud aja ja tasu osas, kuid sain väärtusliku õppetunni, et hinnangut ei tohi anda, kui tehtava töö kohta piisavalt infot pole, isegi siis kui klient nõuab seda. Selle asemel tuleb öelda, et vaja on rohkem teavet või hinnanguks on väga suur hinnavahemik, millest ei ole tegelikku kasu.

4.3 Etapp 3

4.3.1 Pidev kommunikatsioon

Kolmandast etapist õppisin, et kommunikeerimist ei ole kunagi üleliia. Kuigi esmapilgul ei paista vajadust olevat koosoleku jaoks, on siiski kasulik pidevalt suhelda, et kõik ühel lehel oleksid. Kui igaüks saab asjadest omamoodi aru ning kui koos

kliendiga pole kirja pandud ühesugune arusaam iga teema kohta, siis on probleemid ja arusaamatused kerged tulema. Oluline on, et mõtted ja kokkulepped oleksid kirja pandud, sest kui juhtubki selline situatsioon, et ootused ja reaalsus ei lähe kokku, siis on võimalik järele vaadata, kas klient ootas midagi muud kui esialgselt kokku lepitud või teenusepakkuja oli üle andnud toote, mis ei vastanud kokkuleppele. Esialgu selliseid vigu juhtuski, kuid hea oli see, et võtsime kliendiga õppust ning parandasime suhtlust omavahel. Parandasime seda nii piisavalt, et mõlemad osapooled olid edaspidi motiveeritud projekti arengust ning tundsid ennast hästi koostööd tehes.

4.3.2 Töö hindamine

Kolmandas etapis oli tehtava töö hindamine edukas. Projekti läbipaistvus paranes võrreldes teise etapiga ning nii klient kui ka teenuspakkuja said täpselt aru, miks ja mille eest makstakse. Hinnatud töö maht oli lähedane reaalsele töömahule, sest eelnev analüüs oli olnud põhjalikum.

4.3.3 Klient jäi rahule

Klient oli rakendusega väga rahul ning soovis seda võimalikult kiiresti kasutusse võtta. Meditsiinitöötajad said rakendust esimest korda patsientide peal katsetada ning leidsid, et see aitaks oluliselt aega kokku hoida ja tööprotsesse parandada. Nii patsientide haldamine kui ka küsimustike täitmine oli töötajate sõnul intuitiivne ja mugav

4.4 Etapp 4

4.4.1 Andmete standardiseerimine

Andmete standardiseerimine oli uus teema mulle ning seetõttu võtsin eeskuju juba töötavatest infosüsteemidest. Eesti E-tervise Sihtasutus kasutab ISO OID standardit andmeväljade nimetamiseks [24] ja ma otsustasin rakenduses andmevälju sarnase metoodikaga nimetada. Seda kas see oli hea otsus selgub alles tulevikus kui rakendust kasutama hakatakse.

4.4.2 Dokumentatsiooni koostamine

Paljude IT projektidega võib juhtuda, et peale poolt aastat ei arendata neid enam edasi ning selle vältimiseks ongi haldusdokumendid [25], mida neljandas etapis koostasime. Kuigi esmalt tunduvad need dokumendid üleliigse bürokraatiana lisavad need

turvatunnet, et asju tehakse ühtemoodi ja mõistetakse ühtemoodi. Seetõttu ei teki mõne aja pärast situatsiooni, et ühte asja on lahendatud erineval viisil või üldse mitte ning saadud tulemust on võimatu kasutada ning keeruline edasi arendada.

4.5 Lisaarendusi ei tehtud

Medic Mobile rakendus on kasutusele võetud ilma lisaarendusi tegemata. See tähendab, et rakenduse koodi on kasutad täies mahus sellisena, nagu tootja on selle välja andnud. Oma koodi kirjutamist on teadlikult välditud võimalikult kaua, sest siis pole vaja olnud üles seada tootmisprotsesse nagu CI/CD ning sellega on hulk aega kokku hoitud. Lisaks on ka tootja versiooniuuendusi lihtsam implementeerida. Kuigi tootja rakendus seab teatud raamid rakenduse kasutusele ning pisiarenduste jaoks on vajadusi tekkinud, on siiski siiani kõik probleemid lahendatud konfigureerimise ja *devops* praktikate abil. Selle abil on jõutud tulemusteni kiiremini ja väiksema vaevaga ning kogu vastutus rakenduse eest on tootja nimel, mille abil on riske maandatud.

4.6 Edasised plaanid

4.6.1 Andmebaasiga sidumine

Järgmiste sammudena on plaanis rakendus siduda VH andmebaasiga, et rakendus saaks pärida patsientide andmeid. VH infosüsteemis on juba olemas palju andmeid, mida oleks vaja küsimustikes kuvada. See muudaks rakenduse kasulikumaks ja mugavamaks töötajate jaoks.

4.6.2 Andmete analüüsimine

Lisaks on ootus, et projektiga seotud andmete visualiseerija saaks kasutada rakenduse andmeid, et luua raporteid analüütikaks. Üks paberküsimustike probleeme on, et andmeid ei saa efektiivselt analüüsida. Projekt on edukas olnud andmete efektiivse analüüsi osas siis kui on valminud raport, mis annab sisukat tagasisidet. Analüüsi abil saaks haigla teada, millised valdkonnad vajavad rohkem ennetustööd. Kuna haiguste ja traumade ennetamine on palju odavam kui tagajärgedega tegelemine, siis aitaks analüüs haiglal palju raha kokku hoida ja patsientide tervist parandada. Arendatud rakendus on mänginud olulist rolli sisuka raporti loomisel, aidates andmeid koguda.

4.6.3 Arendusprotsessi käivitamine

Mingi hetk on paratamatult vaja hakata Medic Mobile rakendusele lisaarendusi tegema. Sellisel juhul on mõistlik üles seada CI/CD protsess, et koodimuudatusi saaks paigaldada sagedamini ja usaldusväärsemalt. See toetab ka agiilset lähenemist, et tööd tehakse väiksemate juppide kaupa [26]. Lisaks tuleks üles seada ka repositoorium koodi hoiustamiseks ja versioneerimiseks.

5 Kokkuvõte

Viljandi haiglale sai tehtud digitaalne küsimustike täitmise platvorm. Projekti võib lugeda õnnestunuks, sest kaks probleemi kolmest on leidnud lahenduse ning potentsiaalselt aitab rakendus ka kolmandale probleemile lahenduse leida. Töötajate tööprotsessid paranevad märgatavalt kui ei tule tegeleda enam paberimajandusega ja käekirja probleemidega. Küsimustikud on dünaamilised, seega teevad ka patsiendi täitmiskogemuse paremaks ning välditakse liigset segatust. Lisaks hoiavad töötajad aega kokku sellega, et rakendus teeb automaatseid järeldusi ning nad ei pea küsimustike vastuseid ühekaupa läbi käima. Tehtud järelduste põhjal tehakse ka patisendi automaatne edasi suunamine, mis aitab meditsiinitöötajate tööprotsesse parandada ja kiirendada. Rakendus ei lahenda otseselt suuremahuliste andmete analüüsi probleemi, kuid aitab neid andmeid koguda, mis on eeldus andmete analüüsiks. Järgmiseks sammuks ongi testida kogutud andmete analüüsimist, et valideerida, kas ka kolmas probleem sai VH jaoks parema lahenduse kui varem. Projekti areng on praegu pooleli tootmiskeskkonna üles seadmise juures ning loodetavasti võetakse rakendus varsti VH-s kasutusse.

Kasutatud kirjandus

[1] O. Loit, „Veebimajutus,“ 2018. [Võrgumaterjal]. Available: https://www.veebimajutus.ee/blogi/mvp-toode.

[2] A. Raup, 2014. [Võrgumaterjal]. Available: https://www.meriroos.ee/Stuff/ITIL_V3_Glossary_100313.pdf.

[3] J. Saar, 2011. [Võrgumaterjal]. Available: https://eopearhiiv.edu.ee/e-kursused/eucip/haldus/711_teenustaseme_haldusprotsess1.html.

[4] S. K. White ja L. Greiner, 2019. [Võrgumaterjal]. Available: https://www.cio.com/article/2439501/infrastructure-it-infrastructure-library-itil-definition-and-solutions.html.

[5] M. Rouse, 2020. [Võrgumaterjal]. Available: https://searchdisasterrecovery.techtarget.com/definition/disaster-recovery-plan.

[6] scrum.org, 2020. [Võrgumaterjal]. Available: https://www.scrum.org/resources/what-is-scrum.

[7] LLC, MuleSoft, 2020. [Võrgumaterjal]. Available: https://www.mulesoft.com/resources/api/what-is-an-api.

[8] M. Rouse, 2017. [Võrgumaterjal]. Available: https://searchdatacenter.techtarget.com/definition/configuration-management-database.

[9] B. Kantor, 2018. [Võrgumaterjal]. Available: https://www.cio.com/article/2395825/project-management-how-to-design-a-successful-raci-project-plan.html.

[10] basicdefinitions.org, 2019. [Võrgumaterjal]. Available: https://et.basicdefinitions.org/515-devops.

[11] J. PETTERS, 2020. [Võrgumaterjal]. Available: https://www.varonis.com/blog/what-is-a-proxy-server/.

[12] Ward Cunningham, 2001. [Võrgumaterjal]. Available: http://agilemanifesto.org/iso/et/principles.html.

[13] scrum.org, 2020. [Võrgumaterjal]. Available: https://www.scrum.org/resources/what-is-scrum.

[14] The Motley Fool, 2020. [Võrgumaterjal]. Available: https://www.fool.com/the-blueprint/what-is-jira/.

[15] Red Hat, Inc, 2019. [Võrgumaterjal]. Available: https://opensource.com/resources/what-docker.

[16] Sumo Logic, 2019. [Võrgumaterjal]. Available: https://www.sumologic.com/insight/microservices-architecture-docker-containers/.

[17] Form.io LLC, 2020. [Võrgumaterjal]. Available: https://www.form.io/.

[18] Medic Mobile, Inc, 2020. [Võrgumaterjal]. Available: https://github.com/medic.

[19] A. Koukia, 2017. [Võrgumaterjal]. Available: https://koukia.ca/why-docker-pros-and-cons-949d104478c5.

[20] OWASP Foundation, Inc, 2019. [Võrgumaterjal]. Available: https://owasp.org/www-community/vulnerabilities/Password_Plaintext_Storage.

[21] BrainyMedia Inc, 2020. [Võrgumaterjal]. Available: https://www.brainyquote.com/quotes/bruce_schneier_182286.

[22] PLAN HAMMER, 2008. [Võrgumaterjal]. Available: https://planhammer.io/blog/raci-software/11-benefits-using-raci-project-management-software/.

[23] N. Kokemuller, 2019. [Võrgumaterjal]. Available: https://yourbusiness.azcentral.com/disadvantages-standardization-business-11864.html.

[24] Eesti E-tervise Sihtasutus, 2007. [Võrgumaterjal]. Available: http://pub.e-tervis.ee/manuals/ISO%20OID/1.0/1.0.pdf.

[25] M. Symonds, 2011. [Võrgumaterjal]. Available: https://www.projectsmart.co.uk/15-causes-of-project-failure.php.

[26] I. Sacolick, 2020. [Võrgumaterjal]. Available: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html.