TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andreas Alapert 179227IADB

Ettevõtte pärandsüsteemi integreerimine eksisteerivasse kaasaegsesse süsteemi

Bakalaureusetöö

Juhendaja: German Mumma
MSc

Tallinn 2021

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: Andreas Alapert

17.05.2021

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on autori töökoha pärandsüsteemi ümberkirjutamine ettevõtte uuema infosüsteemi osana, sest pärandsüsteemi edasiarendus samal platvormil ei ole jätkusuutlik. Sealhulgas tuleb arvesse võtta uusi ärilisi nõudmisi.

Analüüsides erinevaid võimalusi pärandsüsteemidest vabanemiseks, selgus, et antud töö käigus lahendatava probleemi jaoks on kõige sobivam teostada üleminek pärandsüsteemilt uuele süsteemile ühe etapina. Seejärel analüüsiti pärandsüsteemi uusi ärilisi nõudmisi ning nendega seotud andmebaasi muudatusi, mida oli vajalik pärandsüsteemi ümberkirjutamisel teostada. Analüüsiti ka rakenduse arhitektuuri, kuhu pärandsüsteemi funktsionaalsus integreerida tuli, et saada ülevaade murekohtadest, millele arenduse käigus tuleb tähelepanu pöörata.

Analüüsile tuginedes teostati vajalikud arendused, millele järgnes põhjalik testimine ning seejärel rakenduse toodangusse panemine.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 6 peatükki, 12 joonist, 1 tabelit.

Abstract

Integrating a Company Legacy System Into an Existing Modernized System

The aim of this thesis is to rewrite author's workplace legacy system as part of a newer system, because development on the current legacy platform is not sustainable. Also the new business requirements must be taken into account.

When analyzing the possibilities to retire the legacy systems, author found that most suitable solution for the current problem would be migrating the whole legacy system functionality at once. After that, new business requirements and their impact to database were analyzed. In addition, architecture of the newer system, where the legacy system functionality would be migrated to was examined to have an overview of the problems with the newer system.

Based on the analysis the required functionality was developed and thoroughly tested. The end result was an application running in the production.

The thesis is in Estonian and contains 24 pages of text, 6 chapters, 12 figures, 1 table.

Lühendite ja mõistete sõnastik

API	Rakendusliides, *application programming interface* Rakendusliides võimaldab reeglistikule vastaval rakendusprogrammil teise programmi teenuseid kasutada
Controller	Kontroller Model-View-Controller arhitektuuri komponent, mis võtab sisendina sisse info ning muudab selle käsklusteks mudeli või vaate jaoks
HTTPS	Hypertext Transfer Protocol Secure Turvaline protokoll andmete edastamiseks arvutivõrkudes
HR	*Human and resources*
JSON	*JavaScript Object Notation* Andmevahetusvorming
Localstorage	Veebilehitseja andmete hoidla
Low-code	Lähenemisviis tarkvaraarenduses, kus rakenduse loomiseks on vajalik vähese koodi kirjutamine
REST	*Representational State Transfer* Kogum arhitektuurilisi piiranguid, mis kirjeldavad ära kuidas veebiteenust ehitada tuleks
Schema	Kogum andmebaasi objekte
SQL	*Structured Query Language* Päringukeel relatsiooniliste andmebaasisüsteemidega suhtlemiseks
YAML	*YAML Ain’t Markup Language* Inimesele loetav andmete jadastamise keel, kasutatakse tavaliselt konfiguratsioonifailide kirjutamiseks.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Ettevõte, kus töö autor töötab, vajab enda ühele sisemisele infosüsteemile suuremaid edasiarendusi. Antud infosüsteem on pärandsüsteem ning selle edasine arendus ei ole jätkusuutlik. Kuna ettevõttel on kasutusel ka uuem infosüsteem, mis osaliselt kasutab pärandsüsteemi andmeid, siis jätkusuutliku arenduse tagamiseks on mõistlik pärandsüsteemi funktsionaalsus integreerida uuemasse süsteemi.

Antud bakalaureusetöö eesmärgiks on ettevõtte pärandsüsteemi ümberkirjutamine uuema süsteemi osana, sealhulgas võetakse arvesse uusi ärilisi nõudmisi.

Esmalt analüüsib töö autor pärandkoodi ja pärandsüsteemidega seotud probleeme ning võimalusi nendest vabanemiseks. Seejärel kirjeldab autor enda ettevõtte pärandsüsteemi probleeme. Järgmiseks selgitab eraldi analüütik välja uue süsteemi funktsionaalsed nõuded, viies läbi intervjuud pärandsüsteemi kasutajatega. Selgunud funktsionaalsete nõuete põhjal analüüsib töö autor nõuete mõju eksisteerivale andmemudelile – kuidas ning mida tuleb muuta. Selle põhjal teeb töö autor andmemudelile vajalikud täiendused – loob juurde, muudab olemasolevaid ning eemaldab üleliigsed olemid ning atribuudid. Seejärel analüüsib töö autor rakenduse arhitektuuri kuhu pärandsüsteem integreeritakse, et tekiks ülevaade murekohtadest, millele arenduse käigus tuleb tähelepanu pöörata. Eraldi vaatluse alla võetakse nii ees – kui ka tagarakendus. Analüüsile tuginedes teostatakse vajalikud arendused. Arendus koosneb ees – ja tagarakenduse ning andmebaasi arendusest. Esmalt tuuakse uude süsteemi täiendatud kujul üle pärandsüsteemi funktsionaalsus. Seejärel arendatakse funktsionaalsus, mis pärandsüsteemis puudus. Arendusele järgneb põhjalik testimine ning rakenduse toodangusse panemine.

2 Taust

Antud peatükis tutvustab töö autor pärandkoodi ja pärandsüsteemidega seotud probleeme ning võimalusi nendest vabanemiseks. Töö autor tutvustab probleemi ka enda töökoha näitel.

2.1 Pärandkood

Pärandkood on termin, mille all on enamasti mõeldud koodi, mis on vana, raskesti mõistetav ja mida tarkvaraarendajad kardavad muuta. Koodi kardetakse peamiselt muuta, kuna puudub kindlustunne, et tehtavad muudatused olemasolevat funktsionaalsust mitte soovitud kujul ei muudaks [1]. Seda mõjutavad järgnevad tegurid:

- Ühiktestide olemasolu
- Arendaja, kes tunneb süsteemi põhjalikult, olemasolu
- Arenduseks kasutatavate tehnoloogiate ajakohasus

Nagu selgub, siis ei ole see alati nii üheselt defineeritud ning väga palju sõltub kontekstist. Seega võib pärandkood tähendada ka koodi, mis ei ole ühiktestidega kaetud või mida on arendatud kasutades aegunud tehnoloogiaid. Samuti võib see tähendada koodi, mis on päritud kelleltki teiselt ning mille juures ükski esialgsetest arendajatest enam ei tööta [2] [3].

Paljud asutused üle maailma on jätkuvalt sõltuvad pärandsüsteemidest. Tegelikult leidub palju põhjuseid, miks pärandsüsteemidest võiks vabaneda. Üheks suurimaks probleemiks on süsteemide haldamisega seotud kulud. Suurte süsteemide muutmine on üldiselt keeruline ning ajamahukas. Lisaks ehitatud tarkvarale iganeb ka infrastruktuur, millel süsteemi jooksutatakse. Spetsiaalse riist – ja tarkvara haldamine on kallim kui kaasaegsed pilvetehnoloogiatel põhinevad lahendused. Kui süsteemi on arendatud

kasutades iganenud tehnoloogiaid, siis vastavate oskustega tööliste leidmine on keeruline ning kallis [4].

Kaasaegne tarkvara põhineb palju kolmandate osapoolte *API'de* funktsionaalsusel – näiteks autentimine ja geograafiline asukoht. Arendajad on teinud *API*'de kasutamise kaasaegsetele programmeerimiskeeltele ning raamistikele mugavaks. Pärandsüsteemide ühildamine kolmandate osapoolte *API'dega* nõuab rohkem tööd ning aega ja ei ole tagatud, et integratsioon töötab nii nagu on ette nähtud. Pärandsüsteemidel on ka suurem turvarisk. Mida vanem tarkvara on, seda rohkem aega on ründajatel olnud, et leida süsteemide ning tehnoloogiate nõrku kohti. Samuti võidakse süsteemi jooksutada operatsioonisüsteemil, mille arendaja enam uuendusi ei väljasta ning olemasolevaid turvavigu ei paranda [4].

Ettevõtetel, kes investeerivad aega ja raha pärandsüsteemide ülalhoidmiseks, väheneb innovaatilisus. Konkurentidel, kes kasutavad uuemaid tehnoloogiaid, on seega eelis – vähenevad süsteemide haldamisega seotud kulud ning uusi lahendusi on võimalik luua kiiremini. Lisaks eelnevale tuleb arvesse võtta ka rakenduste disaini ning kasutatavust. Halb kasutajakogemus on üks peamistest põhjustest, miks kasutajad rakenduste kasutamise lõpetavad. Kõik toimingud peaksid olema võimalikult lihtsalt ja arusaadavalt teostatavad [4] - [6].

2.2 Võimalused pärandsüsteemidest vabanemiseks

2.2.1 Olemasoleva koodibaasi uuendamine

Üheks lahenduseks, kuidas pärandsüsteemidest vabaneda, on uuendada olemasolevat koodibaasi. Lahendus võib esmapilgul tunduda üsna mõistlik, kuid tegelikkuses võib antud lähenemisega tekkida üksjagu probleeme. Esiteks tasub arvesse võtta arenduseks kasutatud tehnoloogiate vanust ning teha selgeks, kas antud tehnoloogiad on veel ajakohased. Mitte ajakohaste tehnoloogiate õppimine ei pruugi pikas perspektiivis olla mõistlik ning nendega töötamine võib olla keerulisem kui uuemate lahendustega [7].

Kindlasti on üheks märksõnaks ühiktestid. Ühiktestide puudumine tähendab suuremat riski ning ajakulu koodi muutmisel. Selleks, et koodi muutmisel riski maandada, tuleks vastata järgnevatele küsimustele:

- Milliseid muudatusi on vaja teha
- Kuidas saab kindel olla selles, et tehtud muudatused omavad soovitud mõju
- Kuidas saab kindel olla selles, et tehtud muudatused midagi kõrvalist ära ei lõhkunud

Nendele küsimustele vastamine eeldab head arusaamist olemasolevast koodist. Mida suurema süsteemiga on tegemist, seda keerulisem see on. Eriti süvendab probleemi olukord, kus arendustiimis puudub mõni süsteemi esialgsetest arendajatest, kes antud süsteemi põhjalikult tunneb [7] [8].

Palju sõltub ka sellest, kui kvaliteetne on olemasolev koodibaas. Kvaliteetselt kirjutatud koodi puhul on muudatuste tegemiseks vajalike sammude välja mõtlemine küll ajamahukas tegevus, kuid koodi muutmine peaks olema suhteliselt lihtne. Kui süsteemil puuduvad ühiktestid, siis ebakvaliteetsele koodile on ühiktestide kirjutamine raske või lausa võimatu [8].

2.2.2 Pärandsüsteemi funktsionaalsuse järk-järgult ümberkirjutamine

Teine võimalus, kuidas pärandsüsteemidest vabaneda, on vana funktsionaalsuse järk-järgult ümberkirjutamine. See tähendab seda, et lisaks pärandsüsteemile hoitakse seni töös ka uut süsteemi, kuni kogu pärandsüsteemi funktsionaalsus on ümberkirjutatud. Selline lähenemine võimaldab sujuvamat üleminekut peamiselt kahel põhjusel:

1. Kasutajad saavad piiratud määral juba õppida, kuidas uus süsteem toimib
2. Võimaldab arendajatel paremini hoida süsteemi erinevaid komponente üksteisest eraldatuna

Antud lähenemisega tuleb arvesse võtta seda, et osa funktsionaalsust võib olla omavahel tihedalt seotud ning sellised funktsionaalsused tuleb kõik korraga uude süsteemi üle viia [7] [9].

2.2.3 Pärandsüsteemi funktsionaalsuse korraga ümberkirjutamine

Kolmas võimalus pärandsüsteemidest vabanemiseks on vahetada korraga kogu vana platvorm välja uue vastu [7]. Antud lähenemisega tuleb arvestada järgnevate probleemidega:

1. Sõltuvalt süsteemi mastaabist võib selline lähenemine osutuda väga ajakulukaks ning kalliks - näiteks pangasüsteemide nullist kirjutamine võib aega võtta aastaid ning arendusega seotud kulud ulatuvad mitmetesse miljonitesse [7].

2. Uue süsteemi arenduse kestel on tegemist paigalseisuga. See tähendab seda, et uued suuremad muudatused ei ole kättesaadavad enne, kui uus süsteem on toodangusse jõudnud [7].

2.3 Probleemi taust ettevõtte näitel

Ettevõtte, kus töö autor töötab, on Eestis tegelev tarkvaraarendusettevõte. Töötajaid on kokku üle saja. Ettevõte kasutab enda igapäeva töö korraldamiseks kahte ettevõttesisest infosüsteemi. Mõlema süsteemi eesmärgiks on vähendada bürokraatiat.

2.3.1 Siseveeb

Ettevõtte siseveebi näol on tegemist süsteemiga, mis on kasutamiseks mõeldud kõikidele töötajatele ning hõlmab endas järgnevat funktsionaalsust:

- Info leidmine teiste töötajate kohta
- Puhkuste ja lähetuste haldus
- Projekti töötundide sisestamine
- Aastavestluste läbiviimine
- Dokumendihaldus – CV-d ja erinevad sertifikaadid

2.3.2 Sidrun

Tegemist on süsteemiga, mis on mõeldud kasutamiseks projektijuhtidele, finantstöötajatele ja kõrgema taseme juhtidele ning hõlmab endas järgnevat funktsionaalsust:

- Töötajate haldamine
- Projektihaldus – projektide lisamine/muutmine, inimeste planeerimine projektidesse
- Projektidega seotud dokumendihaldus – aktid ja arved
- Finantsraportite genereerimine

Süsteem Sidrun (edaspidi pärandsüsteem) on antud töö kontekstis pärandsüsteem ning seda järgnevatel põhjustel – koodibaas on päritud kelleltki teiselt ning ükski esialgsetest arendajatest seda süsteemi enam ei arenda. Olemasoleva koodi muutmine on keeruline.

Pärandsüsteemi on arendatud kasutades *Oracle APEX't*, mis kujutab endast *low-code* platvormi. Rakenduste arendus toimub kasutades graafilisi tööriistasid ning *Oracle PL / SQL* keelt [10]. Probleem seisneb selles, et pärandsüsteemile on kasutajate soovil vaja luua uusi suuremamahulisi arendusi, kuid praegusel platvormil ei ole see jätkusuutlik ning põhjuseid on mitmeid.

3 Pärandsüsteemi analüüs

Antud peatükis analüüsib töö autor enda töökoha pärandsüsteemi probleeme ning töötab välja lahendused probleemidest vabanemiseks.

3.1 Probleemid pärandsüsteemiga

Esimeseks probleemiks on olemasoleva koodi muutmise keerukus. Esmalt ühiktestide puudumise tõttu, sest puudub ülevaade sellest, kas tehtavad muudatused olemasolevat funktsionaalsust ära ei lõhu. Teisalt seetõttu, et süsteemi eest vastutaval arendustiimil puuduvad teadmised antud arendusplatvormi kohta. Platvormi õppimisel mõtet ei nähta, kuna ettevõte üritab antud platvormi enda uutes projektides vältida.

Teiseks probleemiks on kahe eraldiseisva koodibaasi olemasolu. Kuna siseveebi rakendus laenab arvestataval hulgal andmeid pärandsüsteemi *schemast*, siis tähendab pärandsüsteemis muudatuste tegemine seda, et samaaegselt tuleks muuta ka siseveebi rakenduse koodi. See on arendaja jaoks kindlasti ajakulukam ja keerukam, sest korraga tuleb orienteeruda kahe erineva süsteemi ning tehnoloogia vahel.

Kolmandaks asjaoluks, millele tähelepanu pöörata, on tootjaluku probleem. Arendus *Oracle APEX* platvormiga tähendab seda, et rakendus saab kasutada ainult *Oracle* andmebaasi, mis üldiselt kommertskasutuseks nõuab kallite litsentside soetamist. *Oracle Database Express Edition (*edaspidi *Oracle XE)* võimaldab küll kasutada *Oracle* andmebaasi ilma litsentsitasudeta, kuid sellel on omad piirangud. Hetkel see probleemiks ei ole, kuid ettevõte kus töö autor töötab on võimeline endast välja kasvatama idufirmasid ning seda on ka juba tehtud. Seega võib potentsiaalselt tekkida olukord, kus *Oracle XE* poolt seatavad piirangud osutuvad uuele ettevõttele takistuseks [11] - [13].

Võttes arvesse eelnevalt mainitud probleeme, leiabki töö autor, et ettevõttele on kasulik kirjutada pärandsüsteem ümber siseveebi osana. See peaks hõlbustama edaspidiseid

arendusi ning kasutatavuse vaatenurgast on ühe süsteemi kasutamine kasutaja jaoks mugavam. Samuti ühtlustab see tehnoloogiaprofiili, mis võimaldab vajadusel rohkematel ettevõtte töötajatel arendusest osa võtta. Selle kinnitamiseks on all välja toodud tabel, kus on töötajate oskused tehnoloogiate kaupa, mille nad on siseveebis enda profiili all ära märkinud. Esimesse nelja veergu on paigutatud tehnoloogiad, mis on peamised siseveebi arenduseks kasutatavad tehnoloogiad. Viimases veerus on tehnoloogia, mida on kasutatud pärandsüsteemi arenduseks. Arvesse tuleb võtta seda, et leidub inimesi, kellel mõni tabelis välja toodud oskustest on olemas, kuid mingil põhjusel on see siseveebis märkimata jäänud.

Tabel 1. Töötajate märgitud oskused siseveebis

	Java	**JavaScript**	**Spring Boot**	**ReactJS**	**Oracle Application Express**
Inimeste arv, kellel on siseveebis vastav oskus märgitud	40	35	29	16	9

3.2 Pärandsüsteemi ümberkirjutamine

Pärandsüsteemist vabanemiseks kasutab töö autor eelmises peatükis välja toodud lähenemist, kus üleminek pärandsüsteemilt uuele süsteemile toimub ühe etapina. Seda varianti eelistatakse järk-järgult üleminekule, sest pärandsüsteemis olev funktsionaalsus on omavahel tihedalt seotud ning funktsionaalsuse jagamine pärandsüsteemi ja uue süsteemi vahel oleks ebamugav nii arendajale kui ka lõppkasutajale. Pärandsüsteemi ümberkirjutamine siseveebi osana koosneb järgnevatest etappidest:

- Analüüs (funktsionaalsed nõuded selgitab välja eraldi analüütik)
- Vana funktsionaalsuse ümberkirjutamine täiendatud kujul siseveebi osana
- Uue funktsionaalsuse arendus
- Testimine

- Rakenduse toodangusse panemine

3.3 Uued ärilised nõudmised

3.3.1 Projektid

Pärandsüsteemi üheks peamiseks funktsionaalsuseks on projektihaldus. See võimaldab sisestada erinevaid andmeid projektide kohta (maksumus, olek, kirjeldus, kestvus, dokumentatsioonide asukohad). Samuti saab igasse projekti planeerida töötajaid koos tööaja ning rolliga. Ühest küljest aitab see projektijuhte, sest olemas on selge ülevaade tööjõust. Kuid samuti on sellest funktsionaalsusest kasu ka raamatupidajatele ning kõrgema taseme juhtidele, kellel on ülevaade projektide olekutest ning finantsseisudest.

Pärandsüsteemis on seoses projektihaldusega kaks probleemi, mis tuleb ümberkirjutamisel ära lahendada. Esiteks on pärandsüsteemis võimalik projekti määrata ainult ühesse aastasse. Reaalsuses tekib aga päris palju olukordi kus projekti alg – ja lõppkuupäev jäävad erinevatesse aastatesse. Seega tuleb projekte, mille kestvus on mitmes aastas, sisestada mitu korda ning see on kasutaja jaoks tülikas. Teiseks probleemiks on isikute rollid projektis. Puudub võimalus lisada isikule ühe projekti kohta mitut rolli ning ei ole võimalik projekti sisestada rolli ilma isikuta. Ilma isikuta rollile peab olema võimalik planeerida ka kuu kaupa tunde. See võimaldaks uues süsteemis koostada vaate, kus oleks selge ülevaade mis rollide ning koormustega inimeste järgi on projektijuhil nõudlus. Lisaks peab olema võimalik isikule lisada projekti planeerimisel kommentaari.

Lisaks eelnevalt mainitud vaatele on vajalik koostada vaade ka inimeste projektidesse planeerituse kohta. Sealt peab tulema välja info isikute kohta kuu kaupa – mitu tundi ning kuhu projekti kedagi planeeritud on. See võimaldab projektijuhtidel saada infot vabade töötajate kohta.

Uues süsteemis peab olema näha projekti maksumuse muutmise ajalugu – isik, kes muutis ning uus maksumus. Lisaks projekti maksumusele on vaja tekitada juurde ka väli „Lepingu summa”, mis võib mingitel juhtudel projekti maksumusest erineda. Näiteks juhul kui tegemist on allhankeprojektiga.

Pärandsüsteemis on iga projekt seotud hetkel kindla valdkonnaga. Uues süsteemis on vajalik siduda iga projekt valdkonna asemel kliendihalduriga.

3.3.2 Töötajate haldus

Pärandsüsteemi teiseks peamiseks funktsionaalsuseks on töötajate haldus, millega tegeleb personaliosakond. Uue isiku tööleasumisel sisestatakse süsteemi tema isiklikud andmed ning töökohaga seonduv info (lepingu kestvus, valdkond, koormus). Uus süsteem peab võimaldama näha isikute töökoormuse ning valdkonna muutusi. See on vajalik, kuna üks kaugem planeeritud arendus näeb ette finantsraporti genereerimist raamatupidajatele ning ettevõtte tegevjuhile. Selles raportis on vaja arvesse võtta isiku koormust ning valdkonda kuu täpsusega.

3.4 Andmebaasi muudatuste analüüs

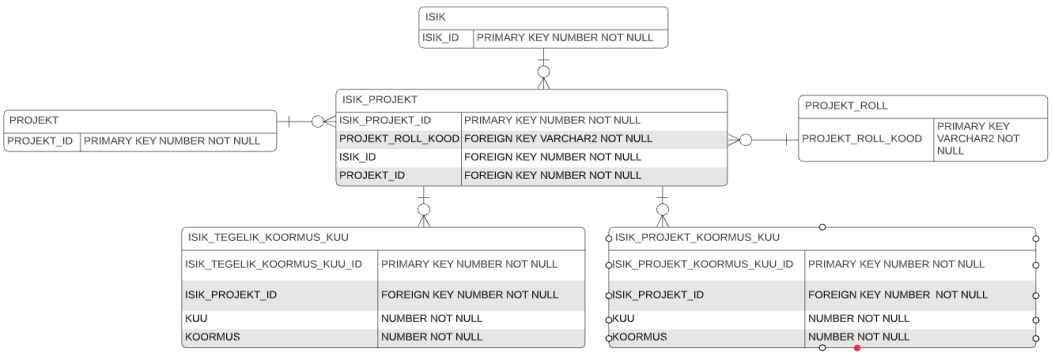
Uute äriliste nõuete tõttu tekib vajadus olemasolevat andmebaasi struktuuri täiendada: tuleb luua uusi olemeid ning muuta olemasolevaid.

3.4.1 Projektihaldusega seotud andmemudeli muudatused

Uute äriliste nõudmistega peab olema võimalik järgnev:

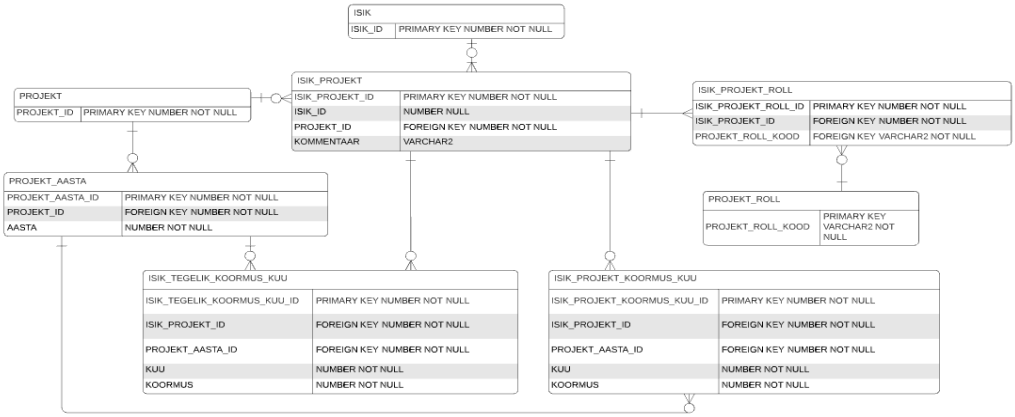
- Projekti kestvust peab olema võimalik määrata mitmesse aastasse
- Projekti saab planeerida rolle ilma isikuteta
- Ühele isikule saab määrata mitu rolli

Järgnevalt on välja toodud olemi-suhte diagramm, mille struktuuri on vaja eelnevalt mainitud nõudmistele kohendada



Joonis 1. Pärandsüsteemi, projektide uutele ärilistele nõudmistele mitte vastav olemi-suhte diagramm

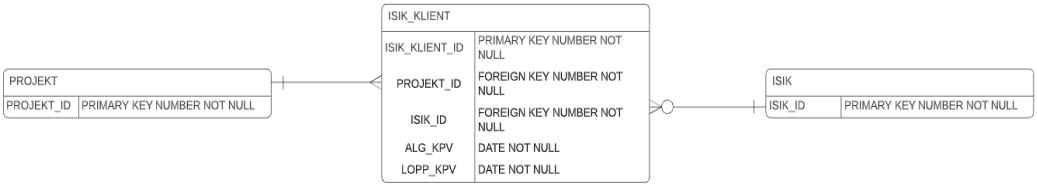
Kuna vana struktuur projektidega seotud uusi ärilisi nõudmisi ei rahulda, siis koostas töö autor uutele nõudmistele vastava olemi-suhte diagrammi.



Joonis 2. Projektide uutele ärilistele nõuetele vastav olemi-suhte diagramm

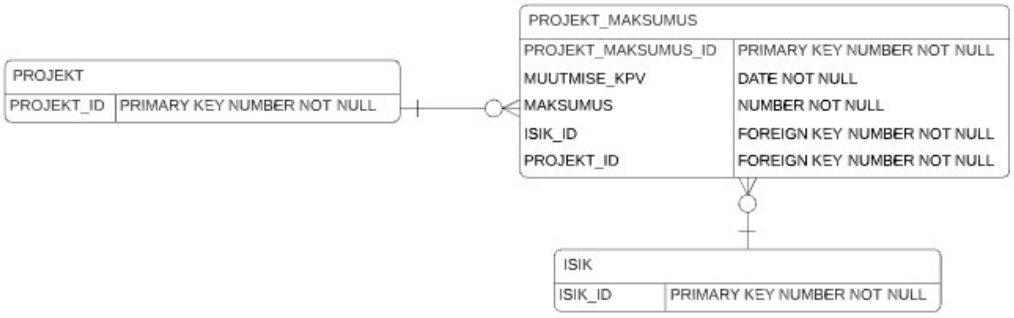
Järgnevalt on ära kirjeldatud kliendihalduri uute äriliste nõudmistega seotud andmemudeli muudatused. Pärandsüsteemi olemi-suhte diagrammis on iga projekt seotud ühe konkreetse valdkonnaga ehk olemis PROJEKT eksisteerib atribuut valdkond_id. Uute nõudmiste järgi on vajalik, et valdkonna asemel tekiks seos

kliendihalduriga ning selle kohta peab olema talletatud ka ajalugu. Uut struktuuri kirjeldab töö autori poolt loodud järgnev olemi-suhte diagramm:



Joonis 3. Kliendihalduri olemi-suhte diagramm

Uute nõudmiste järgi peab olema võimalik projekti maksumuse ajaloo kuvamine. Varem hoiti projekti maksumust tabelis PROJEKT, veerus maksumus. Uut struktuuri kirjeldab töö autori poolt loodud järgnev olemi-suhte diagramm:



Joonis 4. Projekti maksumuse ajaloo olemi-suhte diagramm

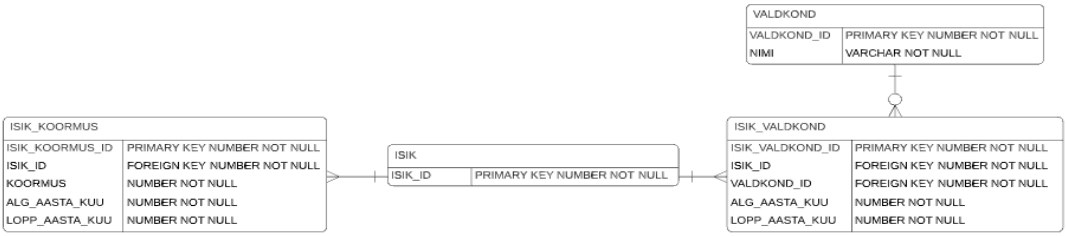
Uus tabel PROJEKT_MAKSUMUS on võimalik täita ajalooliste andmetega, seda tänu PROJEKT logitabeli olemasolule.

3.4.2 Töötajate haldusega seotud andmemudeli muudatused

Uute äriliste nõudmistega peab olema võimalik isiku valdkonna ning töökoormuse ajaloo säilitamine. Isikut ja temaga seotud valdkonda pärandsüsteemis kirjeldab järgnev olemi-suhte diagramm:

Joonis 5. Pärandsüsteemi isiku ja valdkonna vahelist seost kirjeldav olemi-suhte diagramm

Isiku koormuse jaoks on olemis ISIK atribuut „koormus”. Töö autor koostas järgneva olemi-suhte diagrammi, et oleks võimalik nii koormuse kui ka valdkonna ajaloo kuvamine:



Joonis 6. Isiku töökoormuse ja valdkonna olemi-suhte diagramm

Sarnaselt tabelile PROJEKT_MAKSUMUS, on võimalik ka tabelid ISIK_KOORMUS ja ISIK_VALDKOND täita ajalooliste andmetega tänu isiku ajalootabeli olemasolule.

3.4.3 *Schemade* konsolideerimine ning korrastus

Hetkel eksisteerib siseveebi ja pärandsüsteemi poolt kasutatavas andmebaasis kaks eraldi schemat – INTRANET ja SIDRUN. Kuna peale pärandsüsteemi ümberkirjutamist

ei oma kahe *schema* eraldi hoidmine mingit loogilist otstarvet, siis on mõistlik tõsta kaks *schemat* kokku.

Mõlemas *schemas* eksisteerib vanadest aegadest ka tabeleid ning väljasid, mida enam vaja ei lähe. Samuti tekib selliseid väljasid juurde uue baasistruktuuri loomisega. Seega on väga tähtis, et likvideeritaks kõik tabelid ning väljad, mida enam ei kasutata, et vältida arendajates segaduse tekitamist.

3.4.4 Andmemigratsioon

Selleks, et andmed saaksid teisendatud vanalt baasistruktuurilt uuele, on vajalik migratsiooniskriptide kirjutamine. See on koht, millele tuleb kogu arendusprotsessi jooksul kõige rohkem tähelepanu pöörata, sest kui vigaste andmetega rakendus jõuab toodangusse, siis on juba keerulisem veakohta välja selgitada. Tuleb välja töötada tagavaraplaan – mida teha siis kui andmetesse tekivad vead sisse. Seda isegi juhul, kui eelnenud on põhjalik testimine ning kõik tundub korras olevat.

Enne migratsiooniskriptide rakendamist on vaja teha andmetest koopia. Kui nüüd avastatakse andmetes vead, siis on kaks valikuvarianti.

1. Parandada migratsiooniskriptid, rakendada andmetest tehtud koopia, teisendada andmed uuesti parandatud skriptidega

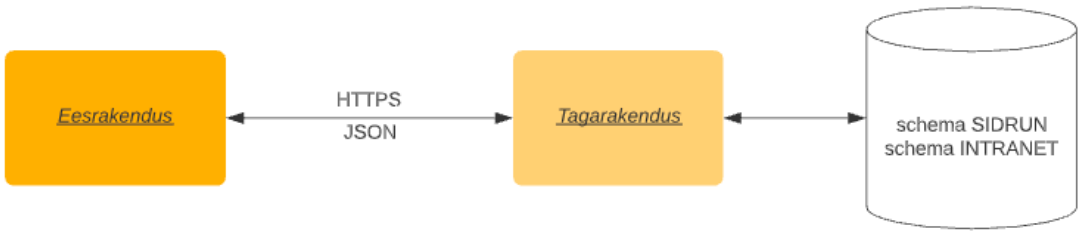
2. Teha andmed käsitsi korda

Kumb variant valida, see sõltub sellest kui kiiresti viga avastatakse. Kui kasutajad on juba suuremas mahus uude süsteemi andmeid sisestanud, siis esimene variant enamasti võimalik ei ole, sest vahepeal sisestatud andmed läheksid kaotsi.

3.5 Siseveebi rakendusega seotud probleemid

Kuna pärandsüsteemi funktsionaalsus tuleb integreerida olemasolevasse siseveebi rakendusse, siis on vajalik analüüsida selle süsteemi arhitektuuri. See on vajalik selleks, et tekiks selge ülevaade murekohtadest, millele tuleks arendusel tähelepanu pöörata.

Siseveeb koosneb eesrakendusest ning tagarakendusest, mis omavahel suhtlemiseks kasutavad *REST* arhitektuuri.



Joonis 7. Siseveebi arhitektuur

3.5.1 Eesrakendus

Siseveebi eesrakendus on oma olemuselt brauseripõhine klientrakendus, mis suhtleb tagarakendusega kasutades *HTTPS* protokolli, andmeid vahetatakse *JSON* vormingus. Peamine asjaolu, millele eesrakenduse arendusel tuleb tähelepanu pöörata, on kasutajaliides. See ei tohiks siseveebi kasutajaliidesest ülemäära palju erineda.

Siseveebi eesrakendust on võimalik kasutada eesti ja inglise keeles. Selle jaoks on olemas mõlemas keeles *JSON* laiendiga tõlkefailide komplektid, mille töötlemiseks on olemas spetsiaalne teek. Selle lahenduse miinuseks on see, et ei ole võimalik mõistlikult tõlkida tagarakendusest alla laetavaid dokumendifaile. See probleem lahendatakse pärandsüsteemi ümberkirjutamisel järgnevalt:

- Tõlkeid hoitakse rakendusserveris *YAML* failidena kujul kategooria_lokalisatsioon.yml
- Tagarakendusse luuakse *controller*, mis väljastab vastavalt päringuga kaasa antud keele parameetrile JSON formaadis tõlked
- Tõlkeid hoitakse veebilehitseja *localstorage's*

- Tõlgetega saadetakse eesrakendusse kaasa ka antud tõlkekomplektist arvutatud räsi, mida hoitakse koos tõlkekomplektiga samuti veebilehitseja *localstorage's*
- Eesrakenduse esmasel laadimisel küsitakse tagarakenduse käest kasutajaliidesest valitud keele tõlkekomplekti kehtiv räsi
- Kui veebilehitseja *localstorage's* hoitav räsi ei ühti kehtiva räsiga, siis küsitakse tagarakendusest vastava keele tõlked uuesti

3.5.2 Tagarakendus

Siseveebi tagarakendus on oma olemuselt *RESTful API*. Rakendus on jagatud järgnevateks kihtideks:

- Esituskiht (inglise keeles *Presentation Layer*)
- Teenusekiht (inglise keeles *Service Layer*)
- Andmete juurdepääsukiht (inglise keeles *Data Access Layer*)

Selline kihiline arhitektuur aitab süsteemi erinevaid komponente üksteisest selgelt eristada. See teeb rakenduse arenduse ning haldamise mugavamaks ja selgemaks.

Tagarakenduse arendusele lisab keerukust olemasoleva andmemudeli uutele ärinõuetele kohandamine. Põhjus seisneb selles, et siseveeb kasutab arvestataval hulgal pärandsüsteemi *schema* andmeid. Selletõttu tuleb hakata muutma päringuid, mis on kirjutatud vana andmebaasi struktuuri jaoks. See muudab arenduse tülikamaks ning ajakulukamaks. Tagarakendusel puuduvad ühiktestid. Rakenduse esialgse arendaja sõnul selletõttu, et nende mitte kirjutamine aitas rakenduse kiiremini toodangusse panna. Sellise lähenemise tõttu on nüüd üsna tõenäoline, et olemasoleva funktsionaalsuse muutmisel hakkavad tekkima vead, mille parandamise tõttu suureneb jällegi projekti ajakulu.

4 Rakenduse realisatsioon

Selles peatükis tutvustab töö autor arenduseks kasutatud tehnoloogiaid, võrdleb uue ja vana süsteemi arhitektuuri ning kirjeldab uue süsteemi testimist ja toodangusse panemist.

4.1 Kasutatavad tehnoloogiad

Kuna pärandsüsteemi funktsionaalsus tuleb integreerida olemasolevasse siseveebi rakendusse, siis selles peatükis kirjeldab töö autor ära siseveebi arenduseks kasutatavad tehnoloogiad. Samuti tutvustab töö autor mõningaid uusi kasutusele võetavaid teeke.

4.1.1 Eesrakendus

Eesrakenduse arenduseks on kasutatud *ReactJS*'i. *ReactJS* on *JavaScripti* teek kasutajaliideste arenduseks, mille loojaks on Facebook. *React* võimaldab luua keerulisi kasutajaliideseid korduv kasutatavatest elementidest, mida kutsutakse komponentideks [14]. Kasutajaliidese arenduse mugavndamiseks on kasutusel *Ant Design* [15] *ning Bootstrap* [16] teegid. Hetkel on vormide halduseks kasutusel teek *formsy-react* [17], kuid siseveebi juhtivarendaja soovil asendatakse see antud töö käigus teegiga *Formik* [18]. Vormide valideerimise lihtsustamiseks võetakse kasutusele teek Yup [19]. Et lihtsustada *Ant Design'i* poolt pakutava vormi komponendi ja *Formiku* ühildamist, võetakse kasutusele teek *formik-antd* [20]. Eesrakenduse sõltuvuste haldamiseks kasutatakse *NPM'i (Node Package Manager)* [21].

4.1.2 Tagarakendus

Siseveebi tagarakenduse arenduseks on kasutusel *Spring Boot* [22] ja Java 8 [23]. Andmete juurdepääsukihis kasutatakse mugavaks päringute tegemiseks MyBatis [24] raamistikku. Tagarakenduse ehitamiseks kasutatakse tööriista *Gradle* [25] ning sõltuvuste paigaldamiseks kasutatakse Maven Central [26] repositooriumit.

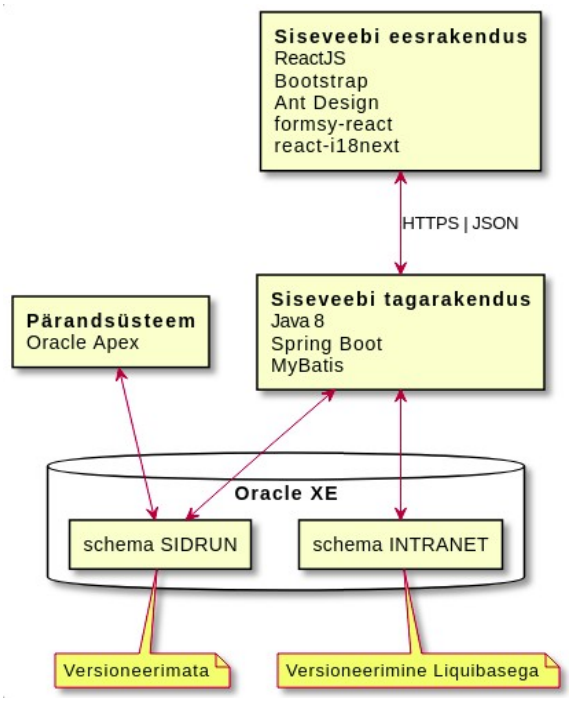
Andmebaasiks on *Oracle XE*. Andmebaasi versioonihalduseks on kasutusel Liquibase [27]. Tagarakenduse arendusel uusi teeke kasutusele ei võeta.

4.1.3 Muu

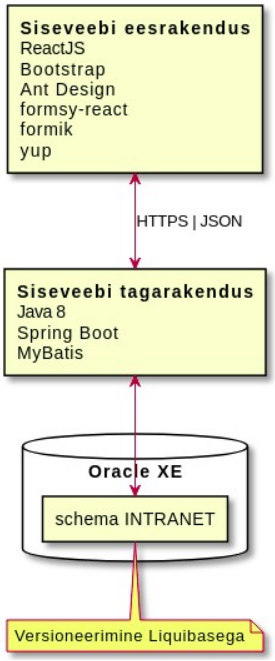
Rakenduse versioonihalduseks on kasutusel Git [28], koodivaramut hoitakse ettevõtte lokaalses GitLab [29] serveris. Ees – ja tagarakendust hoitakse ühes projektis. Java arenduseks kasutatakse ettevõttes *IntelliJ Idea* [30] integreeritud arenduskeskkonda. Eesrakenduse arenduse jaoks kasutab töö autor *Visual Studio Code* [31]. Rakenduse ehitamise ning üles seadmise automatiseerimiseks kasutatakse tööriista *Jenkins* [32].

4.2 Vana ja uue arhitektuuri võrdlus

Järgnevalt on välja toodud süsteemide arhitektuuri joonised enne ning pärast pärandsüsteemist vabanemist.



Joonis 8. Ettevõttesiseste infosüsteemide arhitektuur enne pärandsüsteemist vabanemist



Joonis 9. Arhitektuur pärast pärandsüsteemist vabanemist

Jooniste võrdlusel on selgelt näha, et peale pärandsüsteemist vabanemist on arhitektuur lihtsustunud. Kahe eraldiseisva süsteemi asemel on nüüd üks tervik (Siseveeb) ning andmeid hoitakse ühes versioneeritud *schemas*.

4.3 Testimine ning uue rakenduse toodangusse panemine

Siseveebi rakenduse testimine jaguneb peale uue funktsionaalsuse lisandumist kaheks:

1. Uue funktsionaalsuse testimine
2. Vana siseveebi funktsionaalsuse, mille *SQL* päringud olid koostatud vana andmemudeli põhjal, testimine

Kogu testimine on manuaalne – siseveebi rakendusel puuduvad automaattestid. Pärandsüsteemist üle toodud funktsionaalsusele koheselt automaatteste ei kirjutata – seda siseveebi tooteomaniku otsusel, et rakendus jõuaks planeeritud ajaks toodangusse. Küll on automaattestide kirjutamine järgnevalt kõige prioriteetsem ülesanne, kuid see antud töö skoopi ei kuulu.

Testimist teostab töö autor jooksvalt arenduste käigus, samuti viib põhjalikuma testimise läbi projekti analüütik.

Siseveebi rakendus jookseb kokku kolmes erinevas keskkonnas:

- *Dev*
- *Test*
- *Live*

Enne muudatuste panekut *Live* keskkonda, testitakse kogu protsess nii *Dev* kui *Test* keskkondade peal läbi. Selle jaoks paigaldatakse mõlemasse eelnevalt mainitud keskkonda *Live* andmed, et simuleerida võimalikult täpselt *Live* keskkonna migratsiooniprotsessi. Kui *Dev* ja *Test* keskkonna migratsioonid on edukad ning ilmnenud vead kõrvaldatud, siis võetakse ette migratsiooni protsess *Live* keskkonnaga.

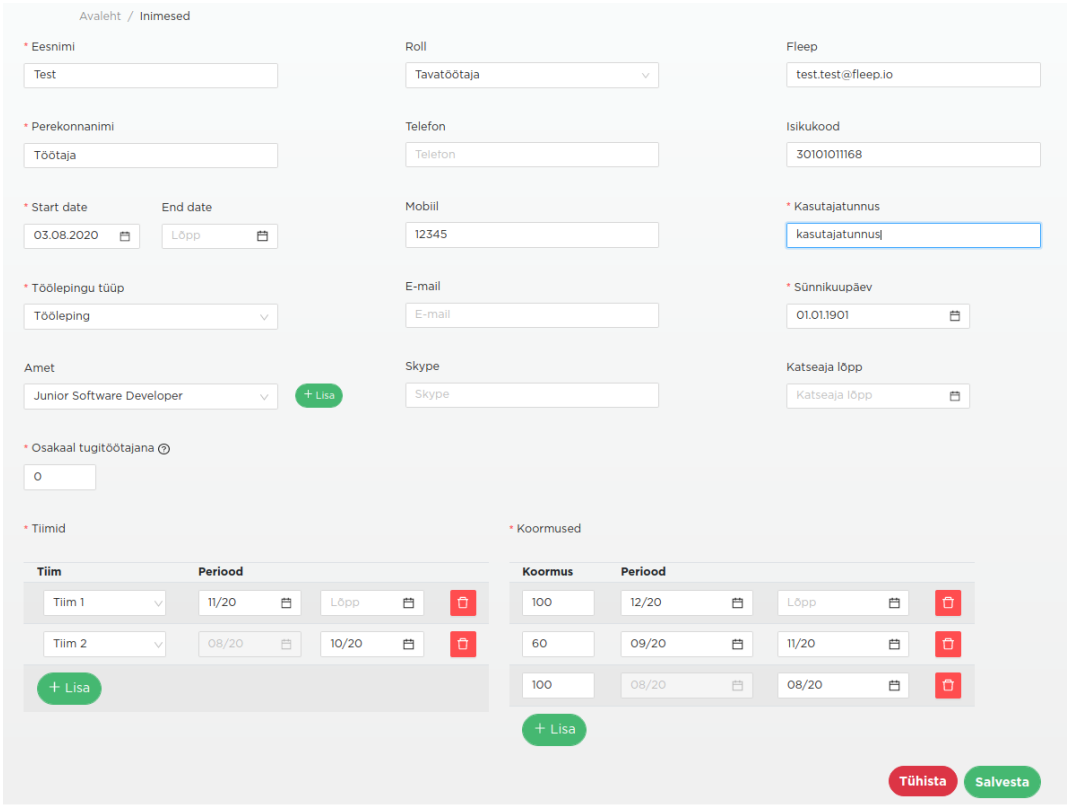
4.4 Loodud funktsionaalsuse tutvustus

Järgnevalt tutvustab töö autor funktsionaalsust, mis sai pärandsüsteemist siseveebi üle toodud.

4.4.1 Töötajate haldus

Töötajate halduse moodul koosneb järgnevatest funktsionaalsustest:

- Töötajate lisamine ja muutmine
- Võrreldes pärandsüsteemiga on nüüd võimalik isikule valdkonda ning töökoormust määrata kuu täpsusega



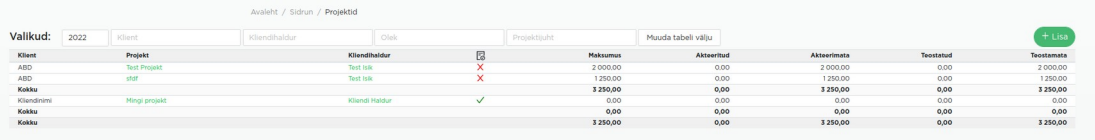
Joonis 10. Isikute haldamise kasutajaliides

4.4.2 Projektid

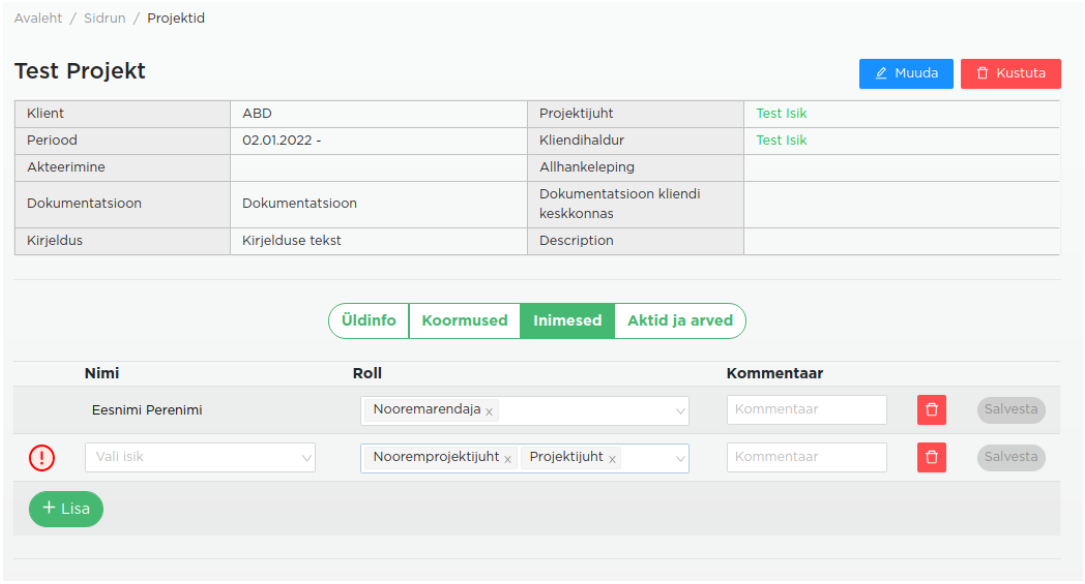
Projektide moodul koosneb järgnevatest funktsionaalsustest:

- Projektide lisamine ja muutmine
- Projektide koondvaade
- Projekti detailvaade
- Võrreldes pärandsüsteemiga on võimalik järgnev:
    - Projekti määramine mitmesse aastasse
    - Võimalik on näha maksumuse muutmise ajalugu

- Iga projekt on valdkonna asemel seotud kliendihalduriga
- Sai loodud kaks uut vaadet. Üks vaade näitab seda, et kuhu projektidesse ning milliste tundide ja rollidega isikud planeeritud on. Teine vaade näitab seda, et millistesse projektidesse mingi rolliga isikutest puudus on. Kaks kirjeldatud vaadet ei ole töö autori loodud.



Joonis 11. Projektide koondvaate kasutajaliides



Joonis 12. Projekti detailvaate kasutajaliides

4.4.3 Aktid ja arved

Aktide ja arvete moodul koosneb järgnevatest funktsionaalsustest:

- Aktide ja arvete lisamine / muutmine projekti detailvaate all
- Aktide ja arvete koondvaade

4.4.4 Projekti töötajate haldus

Projekti töötajate halduse moodul koosneb järgnevatest funktsionaalsustest:

- Töötajate planeerimine projektidesse
    - Roll(id) projektis
    - Töötundide planeerimine kuu kaupa
- Võrreldes pärandsüsteemiga on võimalik järgnev:
    - Ühele isikule on võimalik määrata mitu rolli
    - Võimalik on rollide lisamine ilma isikuta

4.4.5 Kliendihaldurite, valdkondade ja klientide haldus

Pärandsüsteemis puudus liides valdkondade ja klientide haldamiseks – lisamist ja muutmist teostas arendaja käsitsi andmebaasis. Kuna selline lähenemine ei ole jätkusuutlik, siis uude süsteemi sai loodud selle toimingu teostamiseks haldusliides.

Uute äriliste nõudmistega lisandus kliendihalduri mõiste – seega sai loodud sama liides ka kliendihaldurite jaoks.

4.5 Uue süsteemi kasulikkuse hindamine

Pärandsüsteemi ümberkirjutamise kõige suuremaks kasuks on tehnoloogiaprofiili ühtlustumine. Kuna siseveebis kasutatavad tehnoloogiad ning osaliselt ka rakenduse struktuur on valitud ettevõtte teiste uute projektide järgi, siis võrreldes varasemaga on

arendusprotsessi võimalik kaasata rohkem arendajaid. Lisaks on võimalik jagada koodi mingil määral erinevate ettevõtte poolt loodavate rakenduste vahel. Näiteks on peatükis 3.5.1 kirjeldatud tõlkelahendus osaliselt laenatud ühest teisest ettevõtte koodivaramust.

Kui arendaja kasutatavaid tehnoloogiaid juba põhjalikult tunneb, siis on arendusprotsess ka kiirem. Töö autor saab oma teiste projektide kogemuse põhjal väita, et ühtne tehnoloogiaprofiil on väga oluline aspekt, eriti kui alustada töötamist võõra koodibaasiga. See võimaldab paremini lugeda olemasolevat koodi ning lihtsustub ka suhtlus teiste arendajatega.

Teiseks kasuks uue süsteemi puhul on kasutajamugavus. Kasutaja jaoks on ühe süsteemi kasutamine lihtsam. Kui varem tekkis pärandsüsteemiga probleeme või küsimusi, siis tihti ei teatud kelle poole pöörduda. Küsimused suunati enamasti siseveebi arendajatele, kes antud süsteemi ei tundnud ning seega info liikus aeglaselt. Pärast pärandsüsteemist vabanemist on asi lihtne – kõikide probleemidega tegelevad siseveebi arendajad.

5 Võimalikud edasiarendused

Antud töö käigus jäi realiseerimata ettevõtte finantsraporti genereerimine. See ülesanne nõuab enda keerukuse tõttu väga põhjalikku analüüsi ning on planeeritud järgmiseks suureks arenduseks. Samuti on olnud juttu integratsioonist mõne *HR* tarkvaraga. Kuna siseveebi arendavad enda ettevõtte töötajad, siis on kasutajatel väga mugav arendajatega ühendust võtta ning uusi ideid ja parandusi välja pakkuda.

Töö autori hinnangul on kõige suuremaks prioriteediks rakenduse automaattestidega katmine. Pärandsüsteemi ümberkirjutamisel siseveebi osana sai autor automaattestide puudumise valu tunda, sest siseveebi koodi muutmise tõttu tuli muudatusi käsitsi testida. Paratamatult ei suudetud ja osatud kõiki juhtusid läbi testida ning tekkinud vead ei selgunud koheselt.

Põhjalikumat analüüsi vajaks ka andmebaas – tasub kaaluda *Oracle* väljavahetamist vabavaralise *Postgre* [33] vastu, mida ettevõtte teistes projektides järjest rohkem kasutatakse. See lahendaks ka peatükis 3.1 kirjeldatud probleemi, kus *Oracle XE* poolt seatavate piirangute tõttu ei ole andmebaas skaleeritav.

Väga tähtis on teha kõik selleks, et tulevikus vältida probleemi, kus siseveebi rakendus ise muutub pärandsüsteemiks. Selle jaoks on töö autori hinnangul vaja rakendust pidevalt arenduses hoida, et vältida tehnoloogilise võla kuhjumist.

6 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli autori töökoha pärandsüsteemi ümberkirjutamine ettevõtte uuema infosüsteemi osana. Sealhulgas tuli arvesse võtta uusi ärilisi nõudmisi. Muuta tuli projektide ja töötajate haldusega seonduvat.

Analüüsides erinevaid võimalusi pärandsüsteemidest vabanemiseks, selgus, et antud töö käigus lahendatava probleemi jaoks on kõige sobivam teostada üleminek pärandsüsteemilt uuele süsteemile ühe etapina. Põhjuseks oli see, et pärandsüsteemi funktsionaalsus on omavahel tihedalt seotud ning järk-järgult migratsioon oleks ebamugav nii kasutaja kui ka arendaja jaoks. Seejärel analüüsiti pärandsüsteemi uusi ärilisi nõudmisi ning nendega seotud andmebaasi muudatusi, mida oli vajalik pärandsüsteemi ümberkirjutamisel teostada. Loodi juurde uusi olemeid ja atribuute ning muudeti olemasolevaid. Analüüsiti ka rakenduse arhitektuuri, kuhu pärandsüsteemi funktsionaalsus integreerida tuli.

Analüüsile tuginedes teostati vajalikud arendused millele järgnes põhjalik testimine (seda tehti ka jooksvalt arenduste käigus). Testimine koosnes kahest faasist. Esmalt testiti siseveebi vana funktsionaalsust, mille *SQL* päringud olid koostatud vana andmemudeli põhjal. Sellele järgnes uue funktsionaalsuse testimine. Viimaseks etapiks oli rakenduse toodangusse panemine, enne seda teostati migratsiooniprotsess *Dev* ning *Test* keskkondades.

Tänu pärandsüsteemist vabanemisele ühtlustus tehnoloogiaprofiil. Kuna uuemat süsteemi (siseveeb) on arendatud peamiselt tehnoloogiatega, mida ettevõtte enda uutes projektides kasutab, siis nüüd on vajadusel võimalik arendusest osa võtta rohkematel arendajatel ning uusi arendusi saab luua kiiremini. Üks süsteem kahe eraldiseisva asemel on mugavam ka kasutajale.

Kasutatud kirjandus

[1] Perforce, What Is Legacy Code: 8 Tips For Working With Legacy Code, 2018. [Online] Loetud aadressil: https://www.perforce.com/blog/qac/8-tips-working-legacy-code Kasutatud: 06.04.2021

[2] DevOps Zone, Defining Legacy Code, 2018. [Online] Loetud aadressil: https://dzone.com/articles/defining-legacy-code Kasutatud: 10.03.2021

[3] DaedTech, Learning a Healthy Fear of Legacy Code. [Online] Loetud aadressil: https://daedtech.com/learning-healthy-fear-legacy-code/ Kasutatud: 06.04.2021

[4] AltexSoft, Legacy System Modernization: How to Transform the Enterprise for Digital Future. [Online] Loetud aadressil: https://www.altexsoft.com/whitepapers/legacy-system-modernization-how-to-transform-the-enterprise-for-digital-future/ Kasutatud: 21.03.2021

[5] Medium, Principles to Guide Your UX Modernization, 2018. [Online] Loetud aadressil: https://medium.com/headspring-ux-team/principles-to-guide-your-ux-modernization-d1d7ee56270d Kasutatud: 25.04.2021

[6] Modlogix, Risks and Approaches of Legacy Application UI/UX Modernization, 2020. [Online] Loetud aadressil: https://modlogix.com/blog/risks-and-approaches-of-legacy-application-ui-ux-modernization/ Kasutatud: 25.04.2021

[7] Fintech Finance, Migrating from legacy to new banking systems, 2020. [Online] Loetud aadressil: https://www.fintechf.com/01-news/migrating-from-legacy-to-new-banking-systems/ Kasutatud: 13.03.2021

[8] M.C. Feathers, *Working Effectively with Legacy Code:* Prentice Hall, 2004

[9] Archer, Modernization Strategies for Legacy Systems: Choosing the Right One for Your Project. [Online] Loetud aadressil: https://archer-soft.com/blog/modernization-strategies-legacy-systems-choosing-right-one-your-project Kasutatud: 25.04.2021

[10] Oracle Apex, Apex platform. [Online] Loetud aadressil: https://apex.oracle.com/en/platform/ Kasutatud: 23.03.2021

[11] Oracle, Oracle XE. [Online] Loetud aadressil: https://www.oracle.com/database/technologies/appdev/xe.html Kasutatud: 23.03.2021

[12] Oracle Docs, Oracle Database Express Edition. [Online] Loetud aadressil: https://docs.oracle.com/cd/E17781_01/admin.112/e18585/toc.htm#XEGSG101 Kasutatud: 23.03.2021

[13] Oracle, Oracle Technology Global Price List, 2020. [Online] Loetud aadressil: https://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf Kasutatud: 23.03.2021

[14] ReactJS, Intro to React. [Online] Loetud aadressil: https://reactjs.org/tutorial/tutorial.html Kasutatud: 25.04.2021

[15] Ant Design, Ant Design of React. [Online] Loetud aadressil: https://ant.design/docs/react/introduce Kasutatud: 25.04.2021

[16] GetBootstrap, Introduction. [Online] Loetud aadressil: https://getbootstrap.com/docs/5.0/getting-started/introduction/ Kasutatud: 25.04.2021

[17] NPMjs, formsy-react. [Online] Loetud aadressil: https://www.npmjs.com/package/formsy-react Kasutatud: 25.04.2021

[18] Formik, Overview. [Online] Loetud aadressil: https://formik.org/docs/overview Kasutatud: 25.04.2021

[19] GitHub, yup. [Online] Loetud aadressil: https://github.com/jquense/yup Kasutatud: 25.04.2021

[20] GitHub, formik-antd. [Online] Loetud aadressil: https://github.com/jannikbuschke/formik-antd Kasutatud: 25.04.2021

[21] NPMjs docs, About npm. [Online] Loetud aadressil: https://docs.npmjs.com/about-npm Kasutatud: 25.04.2021

[22] Spring, Spring Boot. [Online] Loetud aadressil: https://spring.io/projects/spring-boot Kasutatud: 25.04.2021

[23] Oracle docs, About the Java Technology. [Online] Loetud aadressil: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html Kasutatud: 29.04.2021

[24] MyBatis, Introduction. [Online] Loetud aadressil: https://mybatis.org/mybatis-3/ Kasutatud: 25.04.2021

[25] Gradle docs, Gradle User Manual. [Online] Loetud aadressil: https://docs.gradle.org/current/userguide/userguide.html Kasutatud: 25.04.2021

[26] Gradle docs, Maven Central Repository. [Online] Loetud aadressil: https://docs.gradle.org/current/userguide/declaring_repositories.html#sub:maven_central Kasutatud: 28.04.2021

[27] Liquibase Docs, What is Liquibase. [Online] Loetud aadressil: https://docs.liquibase.com/home.html Kasutatud: 25.04.2021

[28] Git Scm, Git. [Online] Loetud aadressil: https://git-scm.com/ Kasutatud: 25.04.2021

[29] About GitLab, What is GitLab. [Online] Loetud aadressil: https://about.gitlab.com/what-is-gitlab/ Kasutatud: 28.04.2021

[30] JetBrains, Why IntelliJ IDEA. [Online] Loetud aadressil https://www.jetbrains.com/idea/ Kasutatud: 25.04.2021

[31] Visual Studio Code, Getting Started. [Online] Loetud aadressil: https://code.visualstudio.com/docs Kasutatud: 25.04.2021

[32] Jenkins, Jenkins User Documentation. [Online] Loetud aadressil: https://www.jenkins.io/ Kasutatud: 25.04.2021

[33] PostgreSQL, The World’s Most Advanced Open Source Relational Database. [Online] Loetud aadressil: https://www.postgresql.org/ Kasutatud: 05.05.2021

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

Mina, Andreas Alapert

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Ettevõtte pärandsüsteemi integreerimine eksisteerivasse kaasaegsesse süsteemi" mille juhendaja on German Mumma

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.

17.05.2021

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.