TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ken Lilloja 206828IAIB

VETERINAARKONTROLLIDE MIKROTEENUS KOOS EESRAKENDUSEGA

Bakalaureusetöö

Juhendaja: Gert Kanter
PhD

Tallinn 2025

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: Ken Lilloja

04.06.2025

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua Regionaal- ja Põllumajandusministeeriumile (edaspidi REM), mille allasutusel Põllumajandus- ja Toiduametil (edaspidi PTA) on seadusest tulenev kohustus kontrollida inimtoiduks mõeldud liha ohutust ja kvaliteeti, uus töötav veterinaarkontrollide mikroteenus PTA Menetlusinfosüsteemi koos eesrakendusega.

Antud lõputöö tulemusena on valmis mikroteenuse minimaalne töötav versioon, mille abil saab lisada esindusõiguseid tapamajadesse, luua, muuta, otsida veterinaarkontrolle parameetrite alusel ning kinnitada tehtud tapaeelseid ja tapajärgseid kontrolle. Loomade väärkohtlemisele viitavatele pildimaterjalide esitamiseks sai loodud NFS (*Network File System*) server, kuhu talletatakse failid pakitud kujul. Lisaks ka aruandluse jaoks loodud andmete eksport Microsoft Excelisse, mida on vaja saata erinevatel aegadel Statistikaametisse, regiooni juhtidele ning lihakäitlemise spetsialistidele. Eesrakenduse poole pealt valmis kasutajaliides antud mikroteenuse jaoks, mis on modernne ning kasutusel on riiklike e-teenuste jaoks loodud disainisüsteem.

Antud lõputöö laiem eesmärk lõputöö väliselt on teostada ka mitmeid juurdearendusi, nagu näiteks loomaheaolu rikkumiste automaatne saatmine inspektorile. Pikemas perspektiivis on eesmärk automaatselt REM-i Kliendiportaali kaudu suunata kõik loodavad ulukite deklaratsioonid otse MEIS-i ehk digitaliseerida protsess täielikult.

Töö esimeses osas käsitletakse hetkeolukorda, analüüsitakse olemasolevat teenust vanas infosüsteemis ja seletatakse projekti visiooni ja arendusmetoodikat. Seejärel esitletakse rakenduse funktsionaalsust, arutletakse eesmärkide saavutamiseks valitud tehnoloogiaid ning valikute põhjuseid ja tagarakenduse tehnilist lahendust. Viimasena analüüsitakse tehtud töö tulemusi ning defineeritakse järgmised sammud jätkuarendusteks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 7 peatükki, 6 joonist, 0 tabelit.

Abstract

Veterinary Inspection Microservice with Front-end Application

The objective of this bachelor's thesis is to develop a new, functional veterinary inspection microservice for the Ministry of Regional Affairs and Agriculture (hereinafter REM), under which the Agricultural and Food Board (hereinafter PTA) operates. PTA has a legal obligation to ensure the safety and quality of meat intended for human consumption. As a result of this thesis, a minimum viable version of the microservice has been completed, which enables the addition of representation rights in slaughterhouses, as well as the creation, modification, and retrieval of veterinary inspections based on various parameters.

To enable the submission of image material indicating possible animal mistreatment, a Network File System (NFS) server was created, where files are stored in compressed format. Additionally, data export to Microsoft Excel was implemented to support reporting needs, as such reports must be sent periodically to Statistics Estonia, regional managers, and meat processing specialists. On the frontend side, a user interface was developed for the microservice, designed using the national design system for e-services, ensuring a modern and consistent user experience.

Beyond the direct scope of this thesis, the broader goal is to implement additional developments, such as the automatic notification of inspectors about animal welfare violations. In the long term, the aim is to route all game meat declarations directly to the MEIS system via the REM Client Portal, thereby fully digitalizing the process.

The first part of the thesis provides an overview of the current situation, analyzes the existing service in the legacy system, and outlines the project vision and development methodology. This is followed by a presentation of the application's functionalities, a discussion of the technologies chosen to achieve the objectives, and the rationale behind those choices, as well as a description of the backend technical solution. Finally, the results of the work are evaluated, and the next steps for further development are defined.

The thesis is written in Estonian and is 35 pages long, including 7 chapters, 6 figures and 0 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
Autentimine	Protsess kontrollimaks teise olemi identiteedi tõesust
Autoriseerimine	Protsess, mis jagab või keelab olemitele ligipääsu veebiresurssidele
CD	*Continuous Delivery*, pidevintegratsioon, muudatuste pidev tarnimine
CI	*Continuous Integration*, pidevintegratsioon, muudatuste pidev ühildamine
CSRF	*Cross-Site Request Forgery*, ründemetoodika sessiooni äärandamiseks
CSS	*Cascading Style Sheets*, märgistuskeel veebilehtede kujundamiseks
Dockeri konteiner	Tarkvarapakett, mis sisaldab kõike rakenduse käivitamiseks vajalikku
Dockeri kujutis	Juhised, mida kasutatakse Dockeri konteineri loomiseks
EL	Euroopa Liit
Esirakendus	*Frontend*, kasutajale nähtav osa rakendusest
HTML	*HyperText Markup Language*, märgistuskeel veebilehtede sisu defineerimiseks
I/O	*Input, Output*, sisestus ja väljastus
IDE	*Integrated Development Environment*, integreeritud programmeerimiskeskkond
Integratsioonitest	Test, millega kontrollitakse komponentide omavahelist liidestust
JSON	*JavaScript object notation*, lihtne andmevahetusvorming, mis põhineb JavaScripti alamhulgal
JVIS	Järelevalve Infosüsteem, mis on mõeldud PTA ametnikele
JVM	*Java Virtual Machine*, Java virtuaalmasin
Konveier	Sammude kaupa käivitav kood rakenduse testimiseks, ehitamiseks ja tarnimiseks
Lõpp-punkt	Kindla aadressiga liides, mille kaudu tehakse päringuid rakendusega suhtlemiseks
MEIS	PTA Menetlusinfostüsteem

Mikroteenused	Arhitektuuristiil, kus rakendus koosneb omavahel suhtlevatest väikestest teenustest
MS	*Microsoft*, Ameerika Ühendriikide tehnoloogiaarendusettevõte
MVP	*Minimum Viable Product*, minimaalne töötav toode
NFS	*Network File System*, võrgufailisüsteem
NPE	*NullPointerException*, tühja viida erand
Plugin	Pistikprogramm süsteemile täiendava funktsionaalsuse lisamiseks
PTA	Põllumajandus- ja Toiduamet
PVC	*PersistentVolumeClaim*, püsivolüümi taotlus
Raamistik	Platvorm abistamaks arendajat koodi kirjutamisel
REM	Regionaal- ja Põllumajandusministeerium
REST	*Representational State Transfer*, veebirakendustes kasutatav tarkvara arhitektuurilaad
SQL	*Structured Query Language*, standardiseeritud andmebaasipäringukeel
Tagarakendus	*Backend*, serveripoolne tarkvara, mis haldab andmeid ja nende töötlust
Teek	*Library*, korduvkasutamiseks mõeldud funktsioonide ja rutiinide kollektsioon
URL	*Uniform Resource Locator*, internetiaadress
Üksustest	Test, millega kontrollitakse väikese alamprogrammi individuaalset töötamist
XSS	*Cross-site scripting*, ründemetoodika, kus rakendus sunnitakse pahatahtlikku koodi käivitama

Sisukord

Jooniste loetelu

1 Sissejuhatus

Käesolevas peatükis antakse ülevaade valdkondlikust taustast, hetkeolukorrast, töö eesmärgist ning arendusmetoodikast, mille alusel projekt ellu viidi.

1.1 Taust

Liha kontrollimise ja loomade heaolu jälgimise tähtsus on kasvanud koos teadlikkuse suurenemisega toiduohutuse ja eetilise loomakasvatuse valdkonnas. Euroopa Liidu ja Eesti seadusandlus nõuab rangete standardite järgimist, et tagada toiduohutus kogu tarneahelas. [1]

Paljud inimesed on puutunud või puutuvad kokku lihaga toidulaual. PTA-l on seadusest tulenev kohustus kontrollida inimeste toiduks kasutatavat liha. Liha kontrolli teostavad PTA ametnikud – veterinaarid ja veterinaari abilised. Ametnikud teevad liha kohta täheldusi, et oleks fikseeritud, mida kontrolliti ja mida tuvastati. Kontrolli teostatakse kõikide tapmiste korral. Kontrolle on kahte liiki – tapaeelsed ja tapajärgsed kontrollid. Tapajärgsed kontrollid toimuvad alati lihakäitlemisettevõttes (tapamajas). Tapaeelsed kontrollid reeglina toimuvad samuti tapamajas, kuid neid võib teostada ka farmis. Ulukitele teostab ametnik ainult tapajärgse kontrolli. Ulukite laskmisjärgne kontroll teostatakse vastava koolituse läbinud isiku poolt metsas, kes koostab kontrolli kohta ette nähtud deklaratsiooni, mis jõuab tapmajja koos loomaga. [2]

1.2 Hetkeolukord

Hetkel kasutusel olev lahendus on üle kümne aasta vana ning ehitatud REM-i Järelevalve-infosüsteemi (edaspidi JVIS). Tehnoloogia areng ja kasutajate muutuvad vajadused on loonud olukorra, kus süsteem ei vasta enam tänapäevastele nõuetele. Vana arhitektuur ei toeta kaasaegseid töövooge, on kohmakas kasutada ning ei taga piisavat efektiivsust ja kasutusmugavust. Hetkel kasutatav süsteem nõuab ametnikelt märkimisväärselt käsitsi andmete sisestamist, mis suurendab vigade tekkimise riski ja muudab tööaega kulukamaks. Kasutajaliidese vananenud ülesehitus ei ole piisavalt loogiline ning aegunud patoloogiate nimekirjad ei kajasta enam täielikult tänapäevaseid nõudeid ja praktikaid. Lisaks on aja jooksul ametnikel tekkinud selge arusaam, mida ja kuidas muuta töövoos paremaks. Kuna ametnikud peavad järgima ka EL-i nõudeid, tuleb arenduse käigus tagada, et uus lahendus vastaks nendele regulatsioonidele. [3]

1.3 Eesmärk

Lõputöö eesmärk on kavandada ja arendada uus mikroteenus koos eesrakendusega, mis asendab vananenud JVIS-i põhise lahenduse ning parandab süsteemi töökindlust ja kasutajakogemust. Uue teenuse loomisel keskendutakse mitmele olulisele aspektile, sealhulgas funktsionaalsuse laiendamisele, kasutajamugavuse parandamisele, andmete automatiseerimisele ja protsesside digitaliseerimisele.

Esiteks on uue arenduse eesmärk täiendada süsteemi vajalike funktsionaalsuste ja andmeväljadega, mis aitavad ametnikel oma igapäevaseid tööülesandeid tõhusamalt täita. Selleks analüüsitakse olemasolevaid tööprotsesse ja andmevälju, et tuvastada kitsaskohad ning teha vajalikud muudatused. Samuti on oluline arendada uus ja kasutajasõbralikum liides, mis muudab kontrollide sisestamise lihtsamaks ja loogilisemaks. Kuna vana süsteem nõuab palju käsitsi sisestatavaid andmeid, siis uues lahenduses pööratakse erilist tähelepanu andmete eeltäitmisele ja automatiseeritud kontrollmehhanismidele. See aitab vähendada inimlike vigade tekkimise riski ning kiirendab tööprotsessi, võimaldades ametnikel paremini hoida fookusust just töö sisul, mitte tehnilisel poolel.

Lisaks on oluline ajakohastada patoloogiate nimekirjad, et need kajastaksid paremini tänapäevaseid vajadusi ning toetaksid täpsemat andmeanalüüsi ja järelduste tegemist. Vananenud andmestikud võivad põhjustada ebatäpseid analüüse ning raskendada riskihinnangute koostamist. Seetõttu on oluline, et uus süsteem võimaldaks paindlikult ja ajakohaselt hallata patoloogiate ning teiste andmete nimekirju, mis on ametnike töös vajalikud.

Lõputöö käigus viiakse läbi põhjalik analüüs, mille põhjal töötatakse välja uue süsteemi arhitektuur, tehniline lahendus ning minimaalne töötav teenus. Samuti kirjeldatakse arendusprotsessi, testitakse loodud lahendust ning hinnatakse selle töökindlust ja kasutajasõbralikkust. Eesmärk on pakkuda terviklik lahendus, mis ei ole mitte ainult tehniliselt kaasaegne, vaid ka praktiline ja ametnike tööd lihtsustav.

1.4 Arendusmetoodika

Autori vastutada oli mikroteenuse arendamine koos kasutajaliidese täiendamisega, selle juurutamine Kubernetese keskkonda ning integratsioon teiste teenustega. Samuti vastutada nii testimise koordineerimise kui ka versioonihalduse eest, et tagada teenuse sujuv toimimine eri keskkondades. [4]

Arendustiimis töötati aeg-ajalt koos analüütik-disaineriga, kes aitas Figma prototüübi kujundamisel. Tema toel kujunes põhjalik arusaam süsteemi kasutusvoogudest, mille abil sai mikroteenust optimeerida ja võimalikke kitsaskohti ennetada. [5]

Autor töötas agiilse arendusmetoodika järgi kahenädalaste sprintidena, kus ühe sprindi jooksul püüti vähemalt korra kaasata lõppkasutajate esindajaid, et koguda varakult tagasisidet ning vajadusel teha täiendusi. Regulaarne suhtlus lõpp-kliendiga aitas ennetada võimalikke arusaamatusi juba aegsasti ning hoidis arenduse suunatuna tegelikele vajadustele. Lisaks toimusid igal nädalal analüütikuga demod, kus esitleti valminud funktsionaalsusi ning arutati järgmisi arendusetappe. Demo põhjal lisati ka uued ettepanekud arendusplaani, mis tagas pideva iteratiivse arendustöö.

Kuna tegu on uue mikroteenusega, tuli jälgida mikroteenuste vahelisi suhtlusreegleid ja piiranguid, et tagada süsteemi sujuv integreeritus ning optimaalne toimimine kogu infosüsteemis teiste mikroteenustega.

Nii ees- kui ka tagarakenduse arenduseks kasutati IntelliJ IDEA Ultimate koodiredaktorit, kuna sellel on oluliselt suurem funktsionaalsus kui Community versioonil. [6]

2 Ülevaade infosüsteemist

Järgmisena selgitatakse hetkeseisu organisatsioonis ning tuuakse välja senise lahenduse peamised kitsaskohad ja probleemid. Fookuses on olemasoleva teenuse piirangud, mis on tekitanud kasutajatele ebamugavusi, suurendanud töökoormust ning raskendanud süsteemi hooldamist ja andmete kvaliteedi tagamist.

2.1 Organisatsiooni taust

REM on Eesti Vabariigi valitsusasutus, mis vastutab regionaalarengu, maaelu, põllumajanduse ja kalanduse poliitikate kavandamise ning elluviimise eest. Ministeeriumi ülesannete hulka kuuluvad ka looma- ja taimetervise ning toiduohutuse tagamine, samuti põllumajandusteaduse ja -hariduse korraldamine. Ministeeriumi juhib regionaal- ja põllumajandusminister, kellele alluvad kantsler ja asekantslerid. Ministeeriumi valitsemisalas on kaks valitsusasutust: Põllumajandus- ja Toiduamet ning Põllumajanduse Registrite ja Informatsiooni Amet, millest esimesele saab lõputöö käigus loodud uus teenus kasutamiseks. [7]

2.2 Hetkeolukorra kaardistus

PTA kasutab praegu kolme peamist infosüsteemi, milleks on Põllumajandusameti infosüsteem, JVIS ning MEIS. Need süsteemid jagavad omavahel erinevaid valdkondi ja teenuseid, võimaldades ametil tõhusamalt hallata põllumajanduse, toiduohutuse ning järelevalvega seotud protsesse. Lisaks nendele infosüsteemidele on olemas ka Kliendiportaal, mille kaudu saavad tavakasutajad kui ka ettevõtjad esitada erinevaid taotlusi, näiteks mahepõllumajanduse või seemnetaotlusi. [8]

2.2.1 Hetkel toimiva teenuse kitsaskohad

JVIS loodi 2014. aastal, kuid ajas on see järjest suuremaks kasvanud ning paraku kaasneb sellega üha enam jõudluse ja stabiilsusega seotud probleeme, mis on tingitud teekide ning raamistike mitte ajakohasena hoidmisest. Süsteemi ülalpidamine nõuab järjest suuremaid ressursse ning pidevat tähelepanu, mis omakorda suurendab halduskulusid ja IT-osakonna töökoormust. Kasutajad on sageli rahulolematud süsteemi aegluse ja ebastabiilsuse tõttu, mis võib aeglustada nende tööd ja põhjustada olulisi viivitusi järelevalveprotsessides.

Veel üheks oluliseks probleemiks on vananenud kasutajaliides. Praegune süsteem ei ole kohandatud tänapäevastele kasutusstandarditele ning puudub adaptiivne veebilahendus, mis teeb selle kasutamise väiksemate ekraanidega seadmetes, nagu tahvelarvutid ja nutitelefonid, keeruliseks või lausa võimatuks. See omakorda piirab ametnike paindlikkust ja vähendab töö efektiivsust. [9]

2.2.2 Vajadus uue lahenduse järele

Pikaajalise kogemusega ametnikud, kes on JVIS-i kasutanud aastaid, on tänu oma praktilisele töökogemusele teadlikud, millised andmed ja funktsioonid on nende igapäevases töös kõige olulisemad. Samuti oskavad nad hinnata, milliseid uusi tööriistu võiks süsteemi lisada, et parandada tõhusust ja millised olemasolevad funktsioonid on tegelikkuses vähem vajalikud või isegi üleliigsed. Nende teadmised annavad väärtusliku sisendi selle kohta, kuidas loodavat teenust optimeerida nii, et see vastaks paremini reaalsele tööprotsessile ning ei sisaldaks tarbetuid keerukusi. Lisaks on nad kursis ka varasemate probleemidega ja oskavad ennetavalt välja tuua lahendusi, mis aitaksid vältida tulevikus korduvaid vigu või töövoo takistusi.

2.3 Peamised kasutajad

Uue süsteemi peamised kasutajad on PTA ametnikud, kelle tööülesanded on seotud veterinaarkontrolli ja järelevalvega. Kasutajate hulka kuuluvad erinevad spetsialistid ja ametnikud, kellel on vastavalt oma rollile juurdepääs süsteemis vajalikele andmetele ja funktsioonidele. Töö kirjutamise hetke seisuga on antud ametnike ligi 70.

Peamisteks kasutajateks on veterinaarabid, veterinaarametnikud (veterinaarid) ning iga regiooni juhtivspetsialistid, kes vastutavad lihakontrollide eest. Need rollid on otseselt seotud järelevalveprotsessidega, mille käigus tehakse loomade tervisliku seisundi kontrolli, viiakse läbi inspekteerimisi ning dokumenteeritakse vastavad andmed süsteemi kaudu.

Lisaks kuuluvad süsteemi kasutajate hulka kõik PTA ja REM ametnikud ning töölised, kellel on õigus süsteemi sisse logida ja vastavalt oma tööülesannetele andmeid vaadata, sisestada või analüüsida. See hõlmab nii järelevalveametnikke kui ka menetlusspetsialiste, kelle töö nõuab ligipääsu kontrollandmetele ja raportitele. Kuna süsteem on mõeldud laialdaseks kasutamiseks erinevates rollides, peab see pakkuma paindlikku ligipääsuhaldust ning tagama, et iga kasutaja näeb ja saab kasutada vaid oma tööks vajalikke funktsioone. See tagab töö sujuvuse ning süsteemi ning andmete turvalisuse.

2.4 Ootused veterinaarkontrolli mikroteenusele

Veterinaarkontrolli mikroteenuse ootused on põhjalikult läbi arutatud PTA Loomatervise ja -heaolu osakonna ametnikega. Kohtumistel on analüüsitud nende vajadusi ning määratletud peamised funktsionaalsed nõuded, et tagada sujuv ja tõhus tööprotsess. Kuna veterinaarametnikud töötavad peamiselt tapamajades, kus töötempo on kiire ja otsuste langetamine peab olema operatiivne, on oluline, et süsteem toetaks nende igapäevatööd võimalikult intuitiivselt ja tõhusalt.

Oluline on, et kasutajaliides oleks kaasaegne ja loogiline, kasutades Veera disainiraamistikku, mis tagab ühtse ja kasutajasõbraliku keskkonna. Lisaks peab süsteem esile tooma prioriteetsemad (kiireloomulisemad) tegevused, mis aitaksid ametnikel vältida unustamist või oluliste toimingute tähelepanuta jäämist. Selleks võiks lahendus sisaldada näiteks visuaalseid märguandeid ja automaatseid meeldetuletusi, mis juhivad tähelepanu olulistele ülesannetele ning tähtaegadele. [10]

Üks aspekt, millele tuleks ka kindlasti rõhku panna, on logilahenduse korrektne juurutamine, et kõik tegevused oleksid jälgitavad ning vajaliku info saaksid kiiresti kätte IT-osakonna infosüsteemi haldurid, kes abistavad kasutajaid tehniliste probleemide või küsimuste korral. Logide põhjal peab olema võimalik kiiresti tuvastada varasemad toimingud ning vigane koht, et kiiresti võimalikud probleemid lahendada.

Arvestades kettaruumi minimaalset hõivamist ning kiirust, peab failide salvestamine olema kuluefektiivne, et optimeerida nii salvestusmahtu kui ka süsteemi jõudlust. See on eriti oluline suurte andmemahtude puhul, kuna töö käigus kogutakse ja salvestatakse palju dokumentatsiooni, fotosid ning muid tõendusmaterjale. Seetõttu on oluline, et failihaldus oleks struktureeritud ja toetaks kiiret juurdepääsu vajalikule infole.

Lisaks peavad ametnikud kiiresti ja mugavalt saama laadida andmeid alla MS Exceli formaadis. See aitab analüüsida ja töödelda informatsiooni ilma otsepäringuteta andmebaasis, mida saab alla laadida kindlate parameetrite alusel.

Kõik need ootused ja nõuded aitavad kujundada mikroteenuse, mis on töökindel, kasutajasõbralik ning vastab nii praegustele kui ka tulevikus tekkivatele vajadustele, võimaldades samal ajal ametnikel töötada efektiivsemalt ja vältida vigu kiire töötempo juures.

2.5 MEIS-i hetkeseis

MEIS on 2022. aastal esmakordselt kasutusele võetud mikroteenustel baseeruv infosüsteem, mis on loodud kaasaegsete tarkvaraarenduse põhimõtete järgi. Arhitektuur toetub modulaarsele ülesehitusele, mis võimaldab teenuseid arendada, hallata ja laiendada paindlikult ning vajaduspõhiselt.

Hetkel on välja arendatud kaks mikroteenust: antibiootikumide kasutamise jälgimine ning loomade ravimise registreerimine. Kõikidele teenustele on loodud ühtne kasutajaliides, mis tagab sujuva ja loogilise kasutuskogemuse erinevate rollide jaoks. Kasutajaliidese lahendus järgib kaasaegseid disainipõhimõtteid ning on üles ehitatud nii, et see toetaks süsteemi edasist laiendamist ja uute teenuste lisamist.

Ministeeriumi pikaajaline plaan on järk-järgult viia erinevate infosüsteemide teenused üle MEIS-i. See võimaldab ühtlustada protsesse, vähendada eri süsteemide tehnilist võlga ja luua paremini hallatava ning tulevikukindla keskkonna. Käesolev lõputöö on samuti osa sellest protsessist.

3 Loodud mikroteenuse funktsionaalsus

Selles peatükis antakse ülevaade loodud mikroteenuse põhifunktsionaalsusest ning selle töövoogudest, mida ametnikud igapäevases töös kasutavad. Töövoo väljatöötamine oli keerukas ja nõudis koostööd valdkonna spetsialistidega. Kuna protsessi käigus ilmnes vajadus mitmete muudatuste ning täienduste järele, siis tuli ka projekti kestel pisemaid muudatusi teha. Kujundamisel tuli pidevalt arvestada ka kasutajate erinevate privileegidega, määrates täpselt, kes ja milliseid andmeid näha või muuta saab, et tagada süsteemi turvalisus ning rollipõhine ligipääs vastavalt ametnike ülesannetele ja vastutusaladele. Lisaks eeldas funktsionaalsuse kujundamine ka analüütilist mõtlemist. Iga lahenduse puhul tuli hinnata võimalikke alternatiive ning valida välja kõige sobivam lähenemine nii tehnilise teostatavuse kui ka kasutusmugavuse vaates.

3.1 Tegevuskohtade esindusõiguse lisamine

Uues loodud mikroteenuses on oluline funktsionaalsus tegevuskohtade esindusõiguse lisamine. Vastava privileegiga ametnikud saavad määrata konkreetsete tegevuskohtade (tapamajade) juurde isikuid, kellel on õigus seal tööülesandeid täita. See võimaldab tagada, et iga tegevuskoha juures toimetavad ainult volitatud ja vastava õigusega inimesed, suurendades kontrolli ja tööprotsesside läbipaistvust. Pärast seda, kui ametnik on inimese konkreetse tegevuskoha alla lisanud, saab isik end tegevuskoha vaatesse registreerida ning alustada vajalike toimingute teostamist, näiteks loomade kontrolli või aruandlusega seotud tegevusi (vt Lisa 2).

Funktsionaalsuse kavandamisel kaaluti ka alternatiivseid lahendusi, näiteks automaatset õiguste jagamist ametipositsiooni alusel. Lõplikuks valikuks osutus siiski käsitsi määramine, kuna see võimaldab suuremat paindlikkust erinevate olukordade käsitlemisel ning tagab, et volitused antakse ainult konkreetsetele isikutele.

3.2 Veterinaarkontrollide üldotsing

Veterinaarkontrollide üldotsing on loodud kui baasfunktsionaalsus, mis võimaldab kasutajatel kiiresti ja paindlikult otsida teostatud veterinaarkontrolle erinevate parameetrite alusel. Otsingut saab teostada näiteks kontrolli kuupäeva, tegevuskoha, staatuse ja muu asjakohase info põhjal, mis võimaldab kasutajatel kiiresti leida vajalikud andmed tööülesannete täitmiseks või järelevalveks.

Lisaks otsinguvõimalustele on funktsionaalsusesse integreeritud aruandluse eksportimise võimalus. Kasutajad saavad soovitud andmed eksportida Exceli faili, kus on välja toodud kõik vajaminevad andmeväljad edasiseks analüüsiks, töötlemiseks või esitamiseks. Selline lähenemine lihtsustab aruandlust ja võimaldab ametnikel tõhusamalt oma töö tulemusi dokumenteerida ja esitada (vt Lisa 3).

Funktsionaalsuse kavandamisel hinnati erinevaid kasutusstsenaariume, kus vajadus kiiresti leida või esitada spetsiifilist teavet mängib olulist rolli. Kuigi ka varasemas süsteemis eksisteeris teatud määral otsinguvõimalus, oli see kasutajamugavuse ja paindlikkuse poolest ebaefektiivne. Uue süsteemi otsingulahendust täiustati, et võimaldada detailsemat ja mitmekülgsemat filtreerimist, mis aitab kiirendada info leidmist.

Aruandluse eksportimise lahenduse väljatöötamisel kaaluti ka teiste formaatide, näiteks PDF-i kasutamist. Siiski otsustati Exceli kasuks, kuna see võimaldab tabelandmete paindlikku töötlemist ning sobib paremini igapäevaste tööprotsesside ja andmeanalüüsi vajadustega.

3.3 Tegevuskoha vaade

Konkreetse tegevuskoha vaade on loodud selleks, et anda ametnikele kiire ja struktureeritud ülevaade konkreetse tegevuskoha all teostatud veterinaarkontrollidest. Vaates kuvatakse kõik tehtud kontrollid, mis on selgelt eraldatud kontrolli tüübi alusel - eraldi on nähtavad tapaeelsed ja tapajärgsed kontrollid. Prioriteetsuse hindamiseks on kontrollid märgistatud ka eraldi tähisega, võimaldades ametnikel kiiresti tuvastada, millised kontrollid vajavad kiiremat tähelepanu.

Tegevuskoha vaates saavad ametnikud põgusalt tutvuda iga kontrolli olulisema teabega (näiteks kuupäev, loomaliik, ametnik) ilma, et oleks vaja avada kogu detailset infot. Lisaks on loodud mugav funktsionaalsus, mis võimaldab olemasoleva kontrolli andmete põhjal kiiresti luua uue kontrolli, kus teatud andmeväljad kantakse automaatselt üle, vähendades käsitsi sisestamise vajadust ja kiirendades tööprotsessi (vt Lisa 4).

Eriti mugavaks teeb töövoo ka see, et tapaeelse kontrolli andmete pealt saab ühe nupuvajutusega luua vastava tapajärgse kontrolli, mille puhul kantakse vajalikud lähteandmed automaatselt uude kirjesse. See vähendab sisestusvigu ning aitab tagada sujuvama ja loogilisema töökorralduse ametnikele, kes peavad haldama suuri andmemahtusid piiratud aja jooksul.

Funktsionaalsuse kavandamisel lähtuti eelkõige eesmärgist lihtsustada ja kiirendada amet-

nike igapäevast töövoogu, vähendades korduvat andmesisestust ning parandades info leitavust. Lisaks koondab tegevuskoha vaade kogu konkreetse tegevuskohaga seotud info ühte vaatesse, võimaldades ametnikel vältida üldotsingu kasutamist iga kord, kui on vaja kiiret ülevaadet. See toetab töövoogude sujuvust, loob parema orienteeritavuse ning aitab suurendada süsteemi kasutatavust igapäevastes olukordades.

3.4 Veterinaarkontrolli loomine, muutmine, kinnitamine ja tühistamine

Veterinaarkontrolli loomise ja haldamise vaade on disainitud selliselt, et võimaldada kiire ja kasutajasõbralik töövoog kõikide kontrollidega seotud toimingute teostamiseks. Ametnikul on võimalus luua uus veterinaarkontroll, kusjuures andmete sisestamine on tehtud võimalikult tõhusaks tänu automaatse täitmise (*autocomplete*) ja valikmenüü (*select*) funktsionaalsusele. Kasutaja saab ka salvestamisel pidevat tagasisidet teavituste näol (vt Lisa 5).

Kontrolli loomisel saab mugavalt sisestada või muuta olulisi andmeid, nagu dokumentide numbrid, kontrolli kuupäevad, seotud loomade andmed ning vajadusel ka patoloogiliste leidude kirjeldused. Lisaks toetab süsteem failide üleslaadimist - näiteks fotod, laboriaruanded või muud toetavad dokumendid, mis on vajalikud kontrolli täielikuks dokumenteerimiseks.

Iga veterinaarkontrolli juures on kasutajale alati selgelt nähtav kontrolli staatus, mis annab kiire ülevaate, kas tegemist on veel koostamisel oleva, kinnitatud või tühistatud kontrolliga. Vatavalt privileegidele saab ka staatuseid muuta.

Otsus koondada kogu sisestusprotsess ühele lehele põhines soovil vähendada kasutajate navigatsioonikoormust ning võimaldada andmete kiiret ja sujuvat sisestamist. Mitmele vaatele ja sammule jagatud lahendus (nt samm-sammult vorm või vahelehtedega ülesehitus) oleks võimaldanud suuremat loogilist eraldatust, kuid samas ka lisa interaktsioonide arvu, mis teeks töövoo visuaalselt katkendlikumaks. Valitud lahendus ehk ühele lehele koondatud sisestamine koos selge staatuse visuaalse kuvamisega toetab ametnike tööstiili, kus teavet sisestatakse tihti operatiivselt.

3.5 Töölauateavitus avalehel

Ametnike igapäevase töövoo toetamiseks ja tähtsate toimingute õigeaegse täitmise jaoks on loodud avalehe teavituste süsteem. Veterinaarametnikud, kellel on kinnitamata veteri-

naarkontrolle, saavad infosüsteemi sisenedes koheselt nähtava töölauateavituse, mis juhib tähelepanu vajalikele tegevustele.

Teavitus kuvatakse selgelt ja silmapaistvalt juba süsteemi avamisel, vähendades riski, et olulised kinnitamised jäävad märkamata või viibivad. Lisaks informatiivsele märguandele on teavituse juurde lisatud ka mugav otsetee, mis viib kasutaja otse konkreetse kontrolli kinnitamise vaatesse. See vähendab tarbetut klikkimist ja otsimist, võimaldades ametnikul kiiresti ja tõhusalt oma tööülesanded täita (vt Lisa 6).

Alternatiivina kaaluti ka teavituste saatmist e-posti teel. Kuigi e-posti teel saadetud märguanded võivad mõnel juhul aidata tähelepanu juhtida olulistele tegevustele, kaasneks sellega mitmeid puudusi. Esiteks tekiks risk infomüra tekkimiseks, kuna ametnike postkastidesse saabub igapäevaselt palju kirju, siis oluline teavitus võib jääda märkamata. Teiseks tooks see kaasa teise süsteemi vahelepõike, mis ei ole kooskõlas eesmärgiga hoida kasutaja fookus infosüsteemis endas.

4 Tehnoloogiline lahendus

Selles peatükis tuuakse välja analüütilisel teel välja valitud tehnoloogiad ning kavandatud arhitektuur.

4.1 Kasutatud tehnoloogiad

Rakendus on arendatud Java programmeerimiskeeles, mis on üks enimkasutatavaid tehnoloogiaid serveripoolsete rakenduste loomiseks. Java on tuntud oma objektorienteerituse, platvormiüleste võimaluste, skaleeritavuse ja turvalisuse poolest. Lisaks toetab see laia valikut API-sid ja omab tugevat arendajate kogukonda, mis tagab pikaajalise toe ning pideva arenduse. Java valik põhines ka autori varasemal kogemusel ja kompetentsil, mis võimaldab keelt efektiivselt kasutada ning lahendada arenduse käigus tekkivaid probleeme kiiresti ja kvaliteetselt. Samuti on Java kasutamine üks kahest asutuse poolt seatud nõuetest arendustööle. Arutelu on täpsemalt kirjeldatud alapunktis 4.1.1. [11]

Rakenduse arendamisel on kasutatud Spring WebFluxi, mis on osa Spring Frameworkist ja võimaldab mitteblokeerivat, asünkroonset andmetöötlust. Valik on täpsemalt lahti seletatud alapunktis 4.1.2. Spring on saanud Java rakenduste arendamisel tööstusharu standardiks, pakkudes suurt hulka paindlikke ja skaleeritavaid lahendusi. Lisaks WebFluxile kasutatakse rakenduses ka Spring Data R2DBC-d, mis on mõeldud reaktiivseks andmebaasiühenduseks, WebFlux Security turvalisuse tagamiseks ning Spring Boot, mis lihtsustab konfiguratsiooni ja muudab rakenduse haldamise kiiremaks ning töökindlamaks. [12, 13, 14, 15]

Projekti ehitamiseks ja sõltuvuste haldamiseks on kasutusel Gradle, mis võimaldab arendajatel projekti kiiresti ja efektiivselt seadistada ning hallata. Gradle on tuntud oma kiire ehitusprotsessi poolest, kuna see kasutab inkrementaalset kompileerimist, mis tähendab, et ainult muudetud osad kompileeritakse uuesti, mitte kogu projekt. Lisaks on Gradle paindlikum kui mõned vanemad alternatiivid, näiteks Maven, kuna see võimaldab paremini kohandada sõltuvusi ja pluginaid vastavalt projekti vajadustele. Kuigi Maven on samuti väga populaarne ehitustööriist ja seda kasutavad paljud arendajad, on sellel teatud piirangud. Maven järgib ranget konfiguratsioonimudelit, mis muudab selle lihtsaks ja loogiliseks, kuid samas ka vähem paindlikuks keerukamate projektide puhul. Samuti on Mavenis konfiguratsioonifailid sageli mahukamad ja nende haldamine võib olla aeganõudvam. Autoril on varasemalt olnud suurem kogemus Maveniga, kuid projekti spetsiifikast lähtuvalt osutus Gradle siiski sobivamaks valikuks. Gradle võimaldab kiiremat ehitust, tõhusamat

automatiseerimist ja paremat paindlikkust, mistõttu on see kaasaegsete rakenduste puhul eelistatud lahendus. [16, 17]

Rakenduse andmete talletamiseks on valitud PostgreSQL, mis on juba kasutusel ka ülejäänud mikroteenuste andmebaasina. PostgreSQL on tuntud oma ACID-toega transaktsioonide, võimsa päringumootori ning laiendatud JSONB andmetüübi poolest, mis teeb selle sobivaks nii relatsiooniliste kui ka poolstruktureeritud andmete haldamiseks. Lisaks on ka hästi skaleeritav ja töökindel, võimaldades suurt hulka samaaegseid päringuid ilma jõudlust oluliselt mõjutamata. See teeb selle ideaalseks valikuks mikroteenuste arhitektuuris, kus iga teenus võib vajada kiiret ja sõltumatut andmehaldust. Lisaks on ka asutuse teiseks nõudeks just PostgreSQL-i kasutamine lisaks programmeerimiskeele valikule. Autoril on varasem kogemus PostgreSQL-i kasutamisel, mis lihtsustab konfiguratsiooni ja optimeerimist vastavalt rakenduse vajadustele. PostgreSQL-i tugev kogukonna tugi ja pidev areng tagavad selle jäämise usaldusväärseks ja kaasaegseks lahenduseks ka tulevikus, mis on avaliku sektori poolt vaadates margilise tähtsusega. [18]

Rakenduse andmebaasi migratsioonide haldamiseks on valitud Flyway, mis võimaldab skeemi versioonihaldust ja automatiseeritud migratsioone. Kuna süsteem kasutab PostgreSQL-i ja koos sellega ka reaktiivset R2DBC andmebaasiühendust, on oluline, et andmete muudatused oleksid hallatavad usaldusväärse ja skaleeritava tööriistaga. Alternatiivide seas kaaluti ka Liquibase'i, mis on sarnane migratsioonitööriist, pakkudes võimalust hallata skeemi muudatusi XML-i, JSON-i või YAML-i kaudu. Liquibase on väga paindlik ja toetab keerukamaid andmebaasi uuendusi, kuid Flyway eeliseks on lihtsam konfigureerimine ja otsekohene SQL-põhine lähenemine, mis sobib paremini olemasolevasse arendusprotsessi. Samuti oli võimalus kasutada Spring Booti sisseehitatud andmebaasi initsiaatorit. Kuigi see lahendus sobib hästi lihtsate projektide jaoks, ei paku see piisavat versioonihaldust ja kontrollitud migratsioone, mida on vaja suuremas mikroteenuste arhitektuuris. Seetõttu osutus Flyway parimaks valikuks, kuna see on stabiilne, lihtne kasutada ja hästi integreeritud Spring Booti rakendusse, pakkudes samal ajal selget ja turvalist andmebaasi uuenduste haldamist. Flyway erinevate versioonidega on autor kokku puutunud ka varasemalt ning ka mitmetes teistes REM-i infosüsteemides on see kasutusel. [19, 20]

Rakenduses on vaja võimalust töödelda ja eksportida MS Exceli faile, mistõttu on valitud Apache POI teek, mis võimaldab luua, muuta ja lugeda .xlsx faililaiendiga dokumente. Haldusala kasutajad töötavad peamiselt MS Office 2016 lahendusega, seetõttu on mõistlik salvestada failid just .xlsx formaadis, et tagada parim ühilduvus, korrektne vormindus ja probleemivaba kasutajakogemus. Samuti vaadeldi JExcelAPI-d, kuid see toetab vaid vananenud .xls formaati, mis ei ole pikaajalises perspektiivis jätkusuutlik valik ning ka

antud teegi dokumentatsioon ei ole parim. CSV-failid oleksid samuti võimalus andmete lihtsaks eksportimiseks, kuid kuna need ei toeta vorminguid, valemeid ega keerukamaid Exceli funktsioone, ei vasta see kasutajate ootustele. Seetõttu osutus Apache POI parimaks lahenduseks, kuna see võimaldab paindlikult hallata Exceli tabeleid, säilitades samas kasutajatele tuttava vormingu. Lisaks on see laiendatav, stabiilne ja hästi dokumenteeritud, mis tagab pikaajalise töökindluse ja hõlpsa hooldatavuse ka tulevikku silmas pidades. [21, 22, 23]

4.1.1 Java versiooni valik

Arendustöö aluseks on valitud Java 21, mis toob kaasa mitmeid olulisi uuendusi ja täiustusi võrreldes varasemate versioonidega. Ühe märkimisväärse täiendusena toetab Java 21 nimetuid klasse ja eksemplarimeetodeid, mis lihtsustavad koodi kirjutamist ja parandavad selle loetavust. Samuti säilitab Java 21 tugeva tagasiühilduvuse, võimaldades olemasolevatel rakendustel sujuvalt uuemale versioonile üle minna. [24]

Java 21 on pikaajalise toega (LTS) versioon, mis tähendab, et sellele pakutakse pikemaajalist tuge ja värskendusi. See on oluline ettevõtetele ja arendajatele, kes soovivad stabiilset ja usaldusväärset platvormi oma rakenduste arendamiseks ja hooldamiseks. Võrreldes sellega on Java 23 mitte-LTS versioon, mille tugi on lühem ja mis on mõeldud pigem uute funktsioonide kiireks tutvustamiseks ja testimiseks. Seetõttu on Java 21 eelistatud valik, kui eesmärgiks on pikaajaline stabiilsus ja tugi. See tähendab, et võimalikud vead ja probleemid on tõenäoliselt juba avastatud ja lahendatud, mis vähendab arendajate riske ja tagab sujuvama arendusprotsessi. Kuna Java 23 on uuem ja mitte-LTS versioon, võib selle kasutuselevõtt kaasa tuua suuremaid riske seoses võimalike stabiilsusprobleemide ja piiratud toe tõttu. [25]

4.1.2 MVC ja WebFlux lahenduse võrdlus

Arendustöö käigus kaaluti kahte peamist lähenemist veebirakenduste loomiseks: traditsiooniline Spring MVC koos virtuaalsete lõimede kasutamisega ning reaktiivne programmeerimine läbi Spring WebFluxi. Pärast põhjalikku analüüsi otsustati valida reaktiivne lähenemine, tuginedes mitmetele punktidele.

Java 21 tutvustas virtuaalseid lõimesid, mis võimaldavad luua kergekaalulisi lõimesid, mida haldab JVM ise, mitte operatsioonisüsteem. See uuendus parandab rakenduste võimekust hallata suurt hulka samaaegseid päringuid ilma märkimisväärse jõudluse languseta. Virtuaalsed lõimed lihtsustavad mitmelõimeliste rakenduste arendamist, kuna

arendajad saavad kirjutada koodi sünkroonses stiilis, muretsemata lõimede haldamise keerukuse pärast. Testid on näidanud, et virtuaalsete lõimede kasutamine võib pakkuda jõudlust, mis on võrreldav reaktiivse programmeerimisega, eriti väiksema koormuse korral. Näiteks uuringus leiti, et virtuaalsete lõimedega Spring MVC rakendus saavutas sarnase läbilaskevõime kui WebFlux, kuni teatud koormustasemeni.

Spring WebFlux on loodud reaktiivse programmeerimise põhimõtetel, pakkudes mitteblokeerivat sisestust ja väljastust ning võimaldades rakendustel tõhusalt hallata suurt hulka samaaegseid ühendusi. Reaktiivne lähenemine on eriti kasulik, kui rakendus peab töötlema suurt hulka I/O-intensiivseid operatsioone, nagu andmebaasipäringud või väliste teenuste kutsed. Kuigi virtuaalsed lõimed pakuvad märkimisväärseid eeliseid ja lihtsustavad sünkroonse koodi kirjutamist, on reaktiivne lähenemine osutunud tõhusamaks suure koormuse ja I/O-intensiivsete rakenduste puhul. Arvestades projekti ja eeldatavat koormust, valiti Spring WebFlux, et tagada parem skaleeritavus ja jõudlus olenemata kasutajate arvust. Lisaks võimaldab reaktiivne lähenemine paremini hallata ressursikasutust ja pakkuda kasutajatele sujuvamat kogemust ka suurematel kasutusperioodidel. [26, 27]

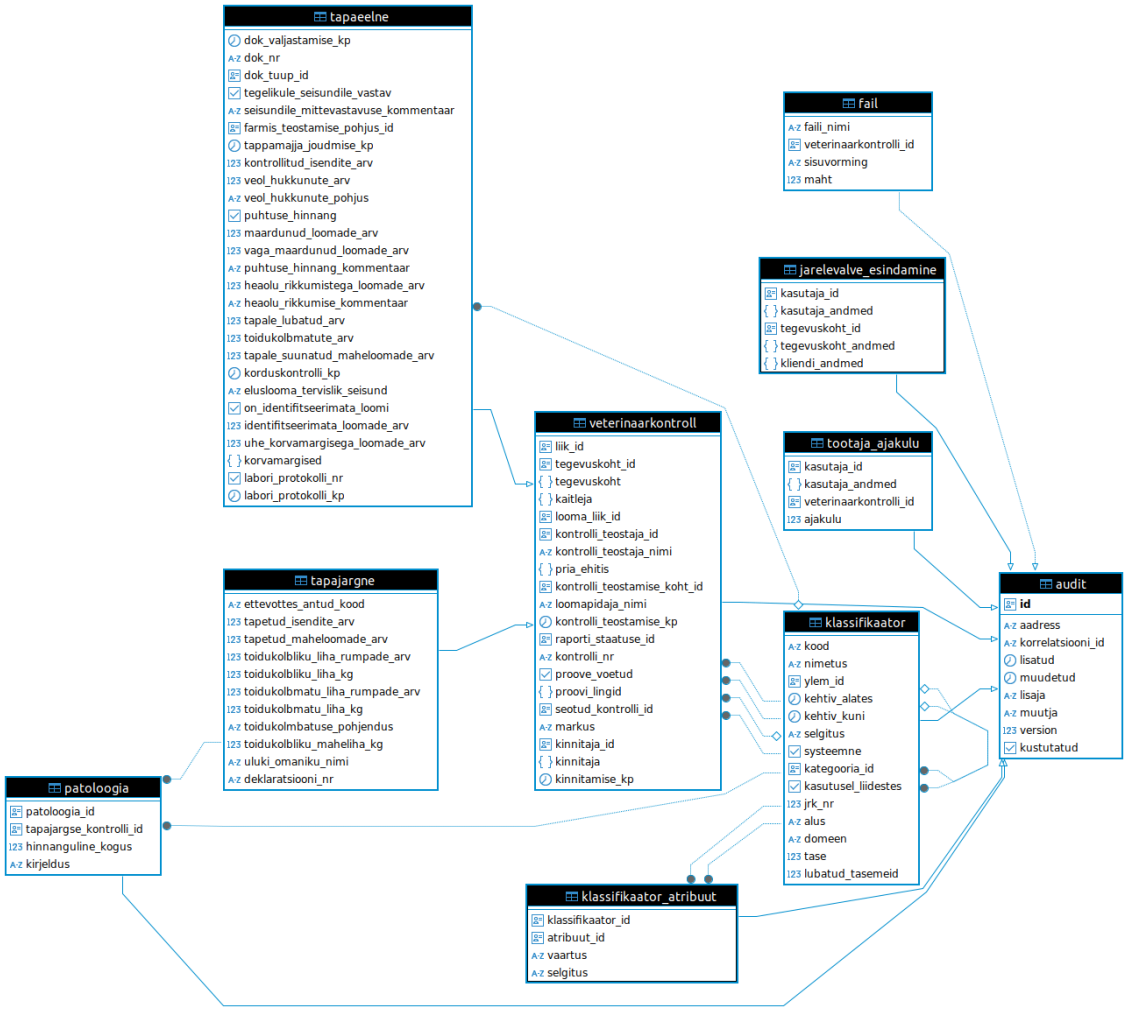
4.2 Andmebaasi arhitektuur

Järgnevalt antakse ülevaade loodud andmebaasi skeemist, mis kajastab tagarakendusega seotud andmebaasi ülesehitust. MVP ulatuses on andmed organiseeritud 15 tabelisse, millest 10 moodustavad põhitabelid ja 5 on abi- ning tugitabelid.

4.2.1 Põhitabelid andmebaasis

Joonisel 1 on kujutatud põhitabelid moodustavad loodud mikroteenuse andmebaasi keskse struktuuri, talletades kogu süsteemi põhitegevuseks vajalikud andmed. Nende hulka kuuluvad näiteks tapaeelse ja tapajärgse kontrolli tabelid, kus hoitakse loomade kontrollide ja seonduvate tulemuste andmeid. Veterinaarkontrolli tabel talletab kõikide kontrollide üldist baasinformatsiooni, sealhulgas seosed tegevuskohtade ja ametnikega, moodustades aluse, millele tapaeelsed ja tapajärgsed kontrollid oma spetsiifilise andmestikuga toetuvad.

Täiendavad põhitabelid nagu patoloogia, fail, järelevalve_esindamine ja tootaja_ajalugu, võimaldavad hallata patoloogilisi leide, kontrollidega seotud faile, esindusõigusi ning kasutajate ajakulu kontrollide kohta. Klassifikaator ja klassifikaator_attribuut tabelid toetavad andmete standardiseeritud kirjeldust ning kategooriate haldamist, samal ajal kui audit tabel dokumenteerib kõiki andmete muudatusi, tagades süsteemi läbipaistvuse ja jälgitavuse.



Joonis 1. *Põhiklasside andmebaasi skeemi osa*

4.2.2 Abi- ja tugitabelid andmebaasis

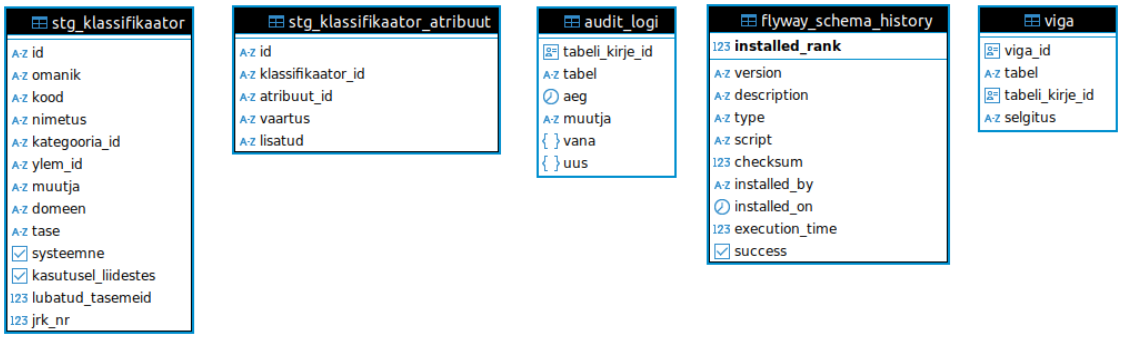
Lisaks põhitabelitele sisaldab loodud andmebaas ka mitmeid abi- ning tugitabeleid, mille eesmärk on toetada süsteemi töökindlust, andmete terviklikkust ja auditeeritavust.

Staging-tabelid (stg_klassifikaator ja stg_klassifikaator_atribuut) on loodud andmete ajutiseks laadimiseks enne lõplikku importimist toodangutabelitesse. Iga migratsiooni käigus laetakse CSV-andmed esmalt nendesse tabelitesse, kus saab neid vajadusel kontrollida, valideerida ja töödelda. See lähenemine võimaldab enne lõplikku salvestamist tagada andmete korrektsuse.

Viga tabelisse logitakse süsteemis tekkivaid kindlaid ja spetsiifilisi vigu, mida autor on pidanud oluliseks ka andmebaasis talletada, mitte ainult logidesse saata. Nii on võimalik vajadusel läbi andmebaasi kiirelt analüüsida vigade mustreid ja hinnata süsteemi töökindlust

pikema perioodi lõikes.

Audit_logi tabeli ülesanne on tagada süsteemi täielik auditeeritavus ja jälgitavus. Selles tabelis säilitatakse iga muudatuse kohta nii vana kui ka uus objekti seis, samuti muudatuse tegija ning toimumise aeg. See võimaldab vajadusel taastada muudatuste ajalugu, kontrollida andmete muutmise protsesse ning toetab järelevalve ja probleemilahenduse tegevusi. Lisaks on ka Flyway poolt automaatselt loodud tabel migratsioonide ajaloo hoidmiseks.



Joonis 2. *Abi- ja tugitabelite andmebaasi skeemi osa*

4.3 Koodi kvaliteedi ja stabiilsuse tagamine

Tarkvaraarenduses on kvaliteedi tagamine üks olulisemaid aspekte, eriti pikaealiste ja pidevalt arenevate süsteemide puhul. Selleks, et tagada koodi veavabadus, puhtus ja hooldatavus, kasutatakse erinevaid staatilise analüüsi ja formaatimise tööriistu. Valitud lahendusteks on Errorprone, SpotBugs ning Spotless, mis aitavad varakult avastada vigu, hoida koodi puhtana ning muidugi ka aidata arendajal vigu tuvastada.

Vigade varajane avastamine on eriti oluline suuremahulistes projektides, kus käsitsi testimine ei suuda kõiki võimalikke olukordi läbi mängida. Google'i Errorprone on analüüsi tööriist, mis töötab Java kompileerimise käigus ning aitab vältida leevendamatuid vigu, näiteks valesti kasutatud võrdlusi, ressursilekkeid või valesid tüübi konversioone. NullAway on spetsiaalne Errorprone laiendus, mis tuvastab võimalikke *NullPointerException*-eid, mis on üks levinumaid vigu Java rakendustes, seega selle varajane tuvastamine aitab vältida ootamatuid tõrkeid toodangukeskkonnas. [28, 29]

Lisaks veatule loogikale on oluline, et kood oleks ka loetav ja ühtlase stiiliga. Selleks kasutatakse Spotless'i, mis automatiseerib koodiformaati ja tagab, et kõik koodiread vastavad samadele stiilistandarditele. Spotless eemaldab ebavajalikud tühikud ja *import*-read, formaatib koodi vastavalt Google Java Format standardile, tagab järjepideva koodistiili, mis on kasulik eriti suuremates projektides, kus võivad aja jooksul tiimiliikmed vahetuda.

Lisaks on kasutusel ka Checkstyle plugin, mille abil saab seada koodi kirjutamise reeglid, mida peab arendamise käigus jälgima, olgu siis selleks kas failinime süntaks või koodirea pikkus. [30, 31]

Lisaks staatilisele analüüsile ja formaatimisele on oluline ka tüüpsisu-koodi vähendamine ja kaasaegsete Java funktsionaalsuste kasutamine. Selleks kombineeritakse Lombok ja Java 21-le iseloomulikud *record* struktuurid, et saavutada kompaktne ja paindlik andmestruktuuride haldamine. Lombok võimaldab annotsioonide abil mugavalt implementeerida *Getter*, *Setter*, *Constructor* ning *Builder* meetodeid, vähendades käsitsi kirjutatava koodi hulka. [32]

API dokumentatsiooni loomine on infosüsteemides olulise tähtusega. Automaatne genereerimine tagab, et süsteem on hästi dokumenteeritud ning lihtsamini mõistetav. Selleks kasutatakse OpenAPI annotatsioone koos SpringDociga, mis loob dünaamiliselt uueneva REST API dokumentatsiooni, vähendades käsitsi kirjutamise vajadust. OpenAPI annotatsioonidega saab määratleda API päringuid, vastuseid ja veakoode, mis muudab API kasutamise lihtsamaks ja arendajatele arusaadavamaks. OpenAPI annotatsioonide tugi on tagatud eraldi teegi swagger-annotations-jakarta kaudu, mis võimaldab täpsemalt kirjeldada API lõpp-punkte, nende päringuid, vastuseid ning võimalikke veakoode. [33, 34]

4.4 Andmestruktuuride töötlemine ning JSON-i salvestamine

JSON tüüpi andmete tõhus töötlemine ja salvestamine on mikroteenustepõhistes arhitektuurides oluline, kuna süsteem peab sageli töötlema dünaamiliselt muutuvaid andmeid. Selleks kasutatakse Jacksoni, mis võimaldab neid andmeid konverteerida Java objektideks ja ka vastupidi. Üheks peamiseks põhjuseks Jacksoni kasutamisel on see, et osa andmeid salvestatakse andmebaasis JSON kujul, et vältida liigseid päringuid teistesse mikroteenustesse. See aitab vähendada süsteemi koormust ning parandada jõudlust ja skaleeritavust. Selliste andmete töötlemiseks on mitmeid alternatiive, kuid Jackson valiti selle kiiruse, paindlikkuse ja laialdase kasutajaskonna tõttu. Alternatiivide hulgas kaaluti Gsoni, mis on lihtsam ning autorile kooliajast tuttav, kuid jääb Jacksonist jõudluses maha ning ei toeta nii hästi komplekssseid Java andmetüüpe. Lisaks on suureks eeliseks dünaamiliste JSON-struktuuride (JsonNode) tugi, mis võimaldab töödelda ka struktuure, mille skeem võib varieeruda. [35]

PostgreSQL toetab salvestamist jsonb tüüpi veergudena, mis võimaldab teha otse päringuid, ilma et peaks neid esmalt Java objektideks deserialiseerima. See vähendab vajadust andmeid pidevalt tõmmata teistest mikroteenustest ning parandab oluliselt süsteemi reakt-

sioonivõimet ja skaleeritavust. Ühe näitena võib tuua tegevuskoha andmed, mis on pärit teisest mikroteenusest. Iga päringu eel selle teenuse poole pöördumine oleks aga ebaefektiivne ja koormaks süsteemi tarbetult. Selle asemel on mõistlik salvestada vajalikud andmed kohalikku andmebaasi, vähendades nii teenustevahelist liiklust ja parandades päringute kiirust.

4.5 Mikroteenuste vaheline suhtlus

Sõnumipõhiseks andmevahetuseks on valitud RabbitMQ, mis võimaldab mikroteenuste vahel asünkroonset, töökindlat ja skaleeritavat suhtlust. RabbitMQ on teiste loodud mikroteenuste puhul juba kasutusel, mistõttu oli see loogiline valik, tagades ühtse ja integreeritud sõnumivahetuse kogu süsteemis. Selle kasutamine võimaldab vältida erinevate sõnumivahendite segadust ja lihtsustab süsteemi hooldamist ning arendust.

Üks RabbitMQ peamistest eelistest on sõnumite vahendamine ja ajutine salvestamine, mis tähendab, et sõnumeid ei pea rakenduse mälus hoidma, vaid need saadetakse vahekihina RabbitMQ kaudu ja vastuvõtja mikroteenus võtab need vastu, kui ta on selleks valmis. See muudab süsteemi väga paindlikuks, kuna saatja ja vastuvõtja ei pea töötama samal ajal – sõnum ei lähe kaduma, kui vastuvõtja on ajutiselt maas. Lisaks toetab erinevaid *routing*-mehhanisme, sealhulgas *direct*, *topic* ja *fanout-exchange*'i, mis võimaldab määrata, kuidas ja millistele teenustele sõnumid edasi saadetakse. [36]

Alternatiivina kaaluti ka Apache Kafkat, mis on suurepärane lahendus suurte andmemahtude töötlemiseks ja reaalajas analüütika jaoks. Kafka pakub logipõhist sõnumi salvestamist, mis tähendab, et sõnumid säilitatakse pikaajaliselt ja neid saab vajadusel hiljem uuesti töödelda. See muudab Kafka väga sobivaks süsteemidele, kus andmete järjepidevus ja ajalooline töötlemine on oluline, näiteks logiandmete või sündmustepõhise vootöötluse puhul. [37]

Siiski osutus RabbitMQ antud projekti jaoks sobivamaks mitmel põhjusel:

1. See on juba kasutusel teistes mikroteenustes, mistõttu tagab see ühtlase ja lihtsama integratsiooni kogu süsteemis.
2. Sobib paremini lühikese elueaga sõnumite jaoks, kus sõnumid ei vaja pikaajalist säilitamist – kui sõnum on korra vastu võetud, ei ole vaja seda uuesti lugeda või säilitada pikaajalises logis, nagu Kafka puhul.
3. Võimaldab sõnumite suunamist ja töökindlat edastamist, hoides ära olukorra, kus andmed kaoksid, kui vastuvõttev teenus pole koheselt saadaval.
4. Lihtsam ja kergemini seadistatav kui Kafka, mis nõuab rohkem ressursse ja keeruka-

mat klastri haldust.

5. Toetab keerukamaid marsruutimisreegleid, mis on kasulikud, kui sama sõnumi peavad vastu võtma erinevad teenused erinevatel tingimustel.

4.6 Andmete valideerimine

Rakenduses on kasutusel Jakarta Validation API, mis võimaldab andmete valideerimist otse mudeliklassides, vähendades sellega liigset käsitsi kirjutatud kontrolliloogikat. Annotatsioonide kasutamine aitab tagada, et süsteemi sisestatavad andmed vastavad määratud reeglitele juba enne nende töötlemist, vältides tarbetuid lisakontrolle teenusekoodis. [38]

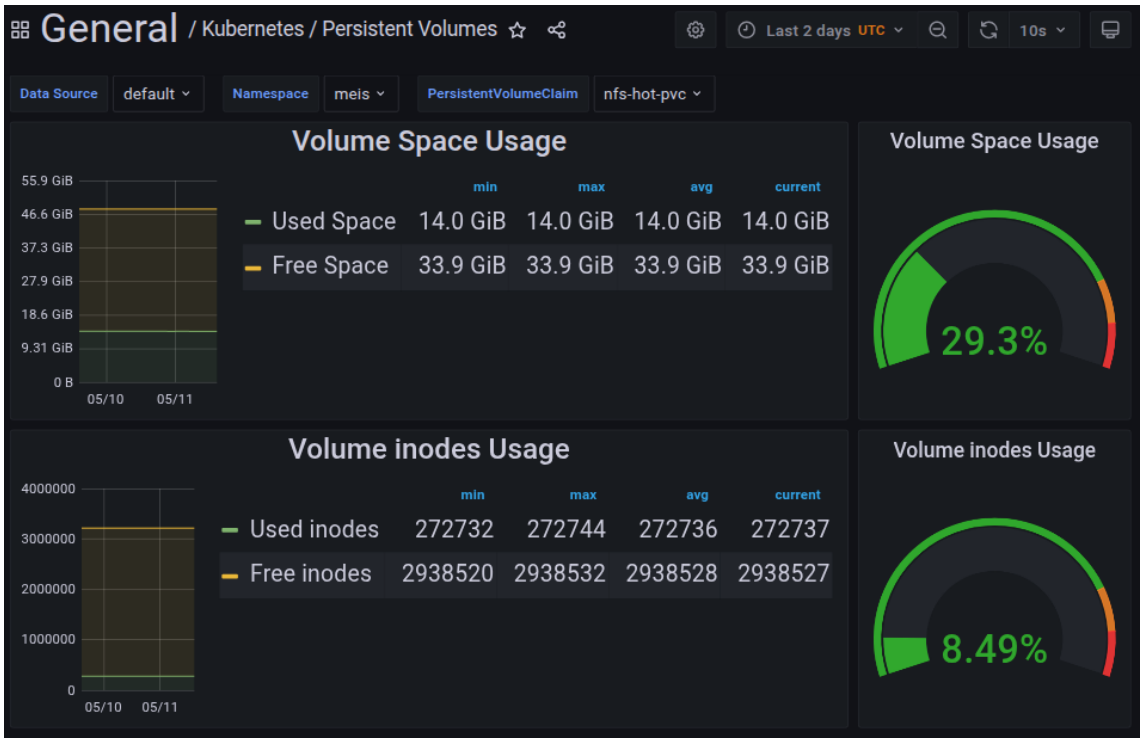
Kuigi valideerimine tehti ka käsitsi, pakub Jakarta Validation API märkimisväärset efektiivsust, kuna see võimaldab deklaratiivset valideerimist otse andmemudelites ilma vajaduseta kirjutada eraldi *if*-lauseid või käsitsi vigu kontrollida. Näiteks *@NotNull* takistab nullväärtuste sisestamist, samas kui *@NotEmpty* hoiab ära tühjade sõnede või kollektsioonide aktsepteerimise.

Valideerimine toimub automaatselt, kui kasutatakse Spring Boot ja WebFluxi kombinatsioonis *@Validated* annotatsiooni kontrolleriklassides, mis tähendab, et sisendandmete kontrollimine toimub enne, kui päring üldse teenuseloogikasse jõuab. See tagab puhtama ja hooldatavama koodi ning aitab vältida vigaste andmete töötlemisest tingitud tõrkeid hilisemas etapis.

4.7 Monitooring

Rakenduse monitoorimise ja jõudluse jälgimise tagamiseks on kasutusel Spring Boot Actuator, mis pakub standardiseeritud viisi süsteemi tervise ja toimivuse jälgimiseks. Lisaks kogutakse ning kuvatakse süsteemiressursside kasutust, sealhulgas JVM mälu, prügikoristuse koormust, lõimekasutust ning ka rakenduse päringute statistikat. Actuatoriga on integreeritud Micrometer, mis võimaldab mõõdikute kogumist sobival kujul sündmuste jälgimise ja hoiatamise tarkvara jaoks. Arutluse all olid Grafana ning Promotheus, millest valiti mõlemad, kuna koostöös pakuvad head terviklahendust ning autor ka neid varasemate projektide peal rakendanud. Graafilise poole pealt on Grafana autori arvates suurepärane, mida ilmestab ka joonis 3, mis kuvab infot loodud volüümi kohta. [39, 40]

Täiendava alternatiivina kaaluti ka Zabbixi kasutamist, mis on ministeeriumis juba laialdaselt kasutusel infrastruktuuris kui ka muudes infosüsteemides. Seetõttu on plaanis integreerida saadud mõõdikud ja logid Zabbixi, et tagada tsentraliseeritud jälgimine ning



Joonis 3. *Kubernetese püsimälu ruumi ja inode'ide kasutuse statistika Grafana kaudu*

võimalikult varajane probleemide tuvastamine. See integratsioon jääb küll lõputöö skoobist välja, kuid seevastu on arendustööde edasiseks faasiks planeeritud. [41]

4.8 Testimine

Rakenduse töökindluse tagamiseks on kasutusel automaatne testimine, mis hõlmab nii üksusteste (*unit tests*) kui ka integratsiooniteste (*integration tests*). Testimise peamine eesmärk on tuvastada vigu varakult, tagada rakenduse korrektne töötamine ning võimaldada turvalisi muudatusi ja arendustööd. Üksustestide läbiviimiseks kasutatakse JUnit 5-e, mis on Java standartne testiraamistik ning pakub kaasaegseid testimisvõimalusi. Sellele on ka mitmeid alternatiive, kuid funktsionaalsuse poolest on nad kõik sarnased, seega saab valida maitse alusel. Lisaks kasutatakse Mockito ja MockWebServeri, et testida HTTP-päringuid ja simulatsioonikeskkondi, vältides testide sõltuvust välistest teenustest. Manuaalset testimist tehti Postmaniga. [42, 43, 44, 45]

Integratsioonitestide puhul kasutatakse testimiseks ka lisaks Testcontainers teeki, et simuleerida täisväärtuslikku rakendusekeskkonda. Testcontainers võimaldab luua ajutisi konteinereid nagu näiteks PostgreSQL ja RabbitMQ, mis võimaldavad realistlikke teststse-naariume ilma käsitsi seadistamata andmebaasi või sõnumivahetust. See tagab, et testitakse rakenduse reaalset käitumist, mitte ainult üksikuid komponente eraldiseisvalt. [46]

Lisaks on kasutusel Reactor Test, mis võimaldab testida WebFluxi ja reaktiivseid andmevooge, mis on eriti oluline asünkroonsete ja mitmekeermeliste operatsioonide testimisel. Kõigi oluliste API-lõpp-punktide testimiseks on loodud täielikud integratsioonitestid, mis valideerivad REST API vastuseid, autentimist ja autoriseerimist, kasutades Spring Security Testi, et simuleerida erinevaid kasutajarolle ja juurdepääsuõigusi. [47]

Lisaks on testid integreeritud ka konveierisse, et tagada nende automaatselt käivitumine igas arendusetapis. See on täpsemalt lahti kirjeldatud peatükis 5.1.

4.9 Turvalisus ja autentimine

MEIS-i süsteemi turvalisuse lahendus põhineb OAuth2 ja JWT-põhisel autentimisel, mis tagab mikroteenuste turvalise ja skaleeritava ligipääsu haldamise. Süsteemi kesksek osaks on värav (Application Gateway / Backend for Frontend), mis suunab kõik tehtavad päringud õigetesse mikroteenustesse ning tagab, et kasutaja identiteet ja õigused on korrektselt kinnitatud. [48]

Kui kasutaja logib MEIS-i sisse, genereeritakse talle JWT, mis sisaldab autentimisandmeid ja õigusi. See identiteeditõend lisatakse kõigisse päringutesse, mida mikroteenused seejärel valideerivad, et kasutaja õigusi kontrollida. Spring Cloud Gateway suudab päringu struktuuri põhjal aru saada, kuhu päring tuleb suunata, mis tagab paindliku, modulaarse ja tõhusa autentimis- ning suunamisprotsessi.

Mikroteenuse turvalisus on implementeeritud Spring Security WebFluxi abil, mis on optimeeritud just reaktiivsete arhitektuurilahenduste jaoks. Peamised turvameetmed hõlmavad järgmist:

- Kõik päringud on autentitud – ainult volitatud kasutajad saavad ligipääsu mikroteenustele
- JWT põhine autentimine – mikroteenused valideerivad kasutaja identiteeti ja õigusi nende poolt edastatud JWT kaudu
- Ohtude vähendamine – CORS ning CSRF on keelatud, et vältida võimalikke turvaauke [49]
- Reaktiivne turvalisus – kogu autentimisprotsess toimub asünkroonselt, et hoida süsteem kiire ja skaleeritav

Kõik API päringud läbivad autentimis- ja autoriseerimismehhanismi, mis:

- Nõuab kehtiva JWT esitamist kõikide päringute puhul, välja arvatud rakenduse

monitooringu ja dokumentatsiooni otspunktid.

- Kasutab turvalist krüpteerimisalgoritmi, et tagada andmete terviklus ja usaldusväärsus.
- Tagab, et ainult volitatud kasutajad pääsevad ligi kindlatele teenustele, piirates juurdepääsu vastavalt kasutajaõigustele ehk privileegidele.

4.10 Keskne failihaldus- ja salvestuslahendus

Süsteemi failihalduse lahendus on loodud keskse jagatud komponendina, millest genereeritakse eraldi JAR-teek, mida kõik mikroteenused saavad kasutada. Selline lähenemine võimaldab ühtset ja standardiseeritud failihaldust üle kogu süsteemi, vähendades dubleeritud koodi ning tagades parema hooldatavuse ja laiendatavuse.

Failide salvestamise ja arhiveerimise protsessis kasutatakse Zstandard (zstd) pakkimisalgoritmi, mis võimaldab saavutada oluliselt parema tihendussuhtarvu ja kiirema pakkimis- või lahtipakkimiskiiruse võrreldes traditsiooniliste meetoditega nagu näiteks Gzip. Kuigi varasemates projektides on autor kasutanud Gzip-pakkimist, otsustati käesolevas lahenduses kasutada Zstandardit, kuna see pakub kaasaegsemat ja tõhusamat lähenemist andmetöötlusele ja salvestusele, eriti suurte failide puhul. Zstandardi eelistest annab hea ülevaate ka Facebook, kes on ise ka antud aloritmi loonud, tehniline blogi, kus rõhutatakse selle kiirust ja väiksemat salvestusmahtu võrreldes teiste populaarsete pakkimisalgoritmidega. [50]

Failide salvestamiseks ja jagatud ligipääsu tagamiseks on loodud NFS server, mis sai üles seatud koostöös Linuxi süsteemiadministraatoriga. Konfigureeriti vajalikud kettapinnad, et võimaldada mikroteenustel failidega turvaliselt ja efektiivselt opereerida.

Edasiarendusena, mis jääb antud lõputöö formaalsest mahust välja, on plaanis rakendada arhiivimismehhanism, mis võimaldab vanemaid, harvemini kasutatavaid faile automaatselt liigutada odavamale ja vähemaktiivsele salvestuskihile, näiteks lindirobotisse või teistesse külmsalvestuslahendustesse. See samm aitaks vähendada kulusid ning optimeerida kettaruumi kasutust, säilitades samas võimaluse failidele vajadusel ligi pääseda. Rakenduslikult võiks selline arhiveerimine toimuda kas ajapõhiselt või vastavalt metaandmetele, näiteks faili loomise kuupäeva või viimase kasutuse alusel.

4.11 Eesrakenduse edasiarendus

MEIS-i kasutajaliidese edasiarendus on samuti olulisel kohal, mis põhineb juba varasemalt loodud Angulari rakendusel. Esialgne arendus pärines ajast, mil Veera disainisüsteem oli alles arenduse algusjärgus, mistõttu tuli mitmed komponendid lahendada käsitsi või ajutiste disainivõtetega. Veera disainisüsteemi ametlik versioon 1.0.0 avaldati 2023. aastal ning see on välja töötatud eesmärgiga ühtlustada riiklike e-teenuste välimust, parandada kasutusmugavust ning tagada vastavus Euroopa Liidu ligipääsetavuse nõuetele. Tegemist on tervikliku disainiraamistikuga, mis sisaldab eelkujundatud ja testitud komponente ning stilistikareegleid, mille kasutamine lihtsustab ja kiirendab arendustööd. [10]

Antud projekti raames on disainisüsteemi komponendid integreeritud olemasolevasse Angulari projekti, mis suurendas rakenduse visuaalset ühtsust, ligipääsetavust ja kasutuskogemuse kvaliteeti. Kasutamine vähendab vajadust juba olemasolevate kasutajaliidese elementide uuesti disainimiseks, mis omakorda hoiab kokku arendusressurssi ning toetab paremini riiklikele suunatud teenuste standardiseerimist.

Kuna kõik vajaminevad kompondid ei ole Veera poolt veel loodud, siis kasutati paralleelselt ka Bootstrap raamistikku – tegemist on ühe enim levinud CSS ja JavaScripti teekidega, mis võimaldab kiiresti luua mobiilisõbralikke ja responsiivseid veebilehti. Bootstrap pakub laia valikut valmis stiilide ja kasutajaliidese elementide komponente (nt nupud, vormid, modaalaknad), mida saab lihtsalt kohandada olemasoleva disainiga. [51]

Veera ja Bootstrapi kooskasutamine võimaldas arendada kasutajaliidest kiiresti, struktureeritult ja ligipääsetavalt, arvestades nii disaini, funktsionaalsuse kui ka riiklike standardite nõudeid. Selline lähenemine võimaldab luua paindliku ja jätkusuutliku eesrakenduse, mis on kergesti laiendatav ja hooldatav tulevikus. Arenduse käigus pandi rõhku ka manuaalsele testimisele väiksemate resolutsiooniga seadmete läbi, et kogu funktsionaalsus oleks erinevate suurustega seadmetes ühtselt kasutatav.

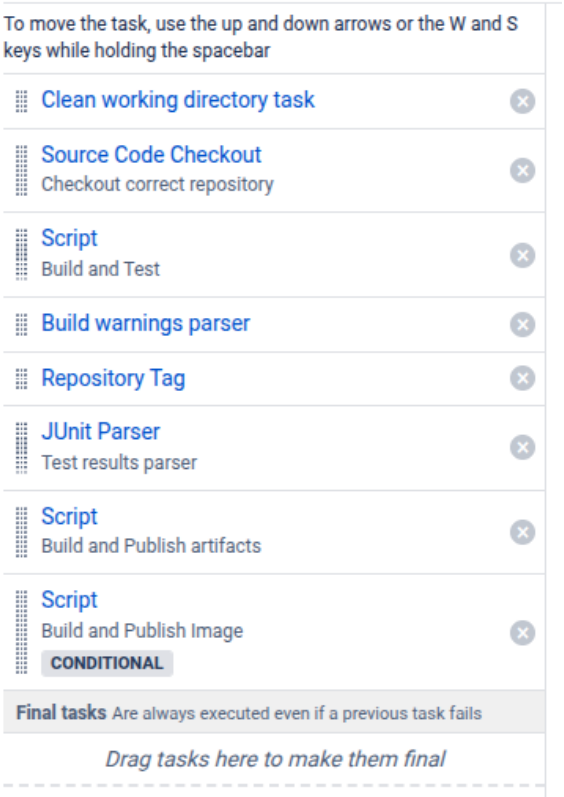
5 Integratsioon

Antud lõputöö skoobis oli lisaks analüüsi ning arendustööle ka tagada sujuv kooditarnete teostamine. Ministeeriumis on kasutusel Atlassiani Bamboo lahendus.

5.1 Automatiseeritud ehitus- ja juurutusprotsess Bamboo keskkonnas

Loodud mikroteenuse CI/CD protsess Bamboo keskkonnas on üles ehitatud struktureeritud ja automatiseeritud töövoona, mis tagab järjepideva ehituse, testimise ja juurutamise. [52]

Järgnevalt arutletakse igast sammust täpsemalt.



Joonis 4. *Loodud mikroteenuse konveier*

1. *Clean working directory task* - See etapp puhastab töökausta enne ülejäänud tegevuste alustamist. Kõik varasemad failid ning kaustad kustutatakse, et tagada iga kord puhas ehituskeskkond.
2. *Source Code Checkout* - Tõmbab alla õige lähtekoodi koodihoidlast vastava valitud haru puhul.

3. *Script – Build and Test* - Käivitab käsitsi määratletud skripti, mis kompileerib lähtekoodi ja viib läbi automaattestid. Kasutatud on käsk *gradlew clean install*.
4. *Build warnings parser* - Analüüsib ehitusprotsessi käigus genereeritud hoiatuste kohta logisid. Selle eesmärk on tuvastada potentsiaalsed probleemid lähtekoodis, mis ei pruugi veel põhjustada ehituse ebaõnnestumist, kuid võivad viidata kvaliteediprobleemidele, mis võivad lähitulevikus probleemiks osutuda.
5. *Repository Tag* - Loob märgise lähtekoodi hoidlas vastava hetke koodiseisuga. Konkreetset versiooni, mida saab vajadusel hiljem reprodutseerida ning on hiljem taasleitav koodihoidlas.
6. *JUnit Parser – Test results parser* - Analüüsib testide tulemusi, et neid visuaalselt esitada CI tööriistas. Võimaldab jälgida, mitu läbis edukalt ning millised testid ebaõnnestusid.
7. *Script – Build and Publish artifacts* - Koostab tarkvara lõplikud artefaktid ja laeb need üles artefaktide hoidlasse. Ministeeriumis on selleks siseseks kasutuseks enda poolt loodud hoidla.
8. *Build and Publish Image* - See samm käivitatakse ainult kindlatel tingimustel, täpsemalt kui Bamboo plaan käivitatakse koos ette antud sildi versiooniga. Ehitatakse valmis Dockeri konteiner ja avaldatakse see konteinerite hoidlas, milleks on eraldi register ministeeriumil.

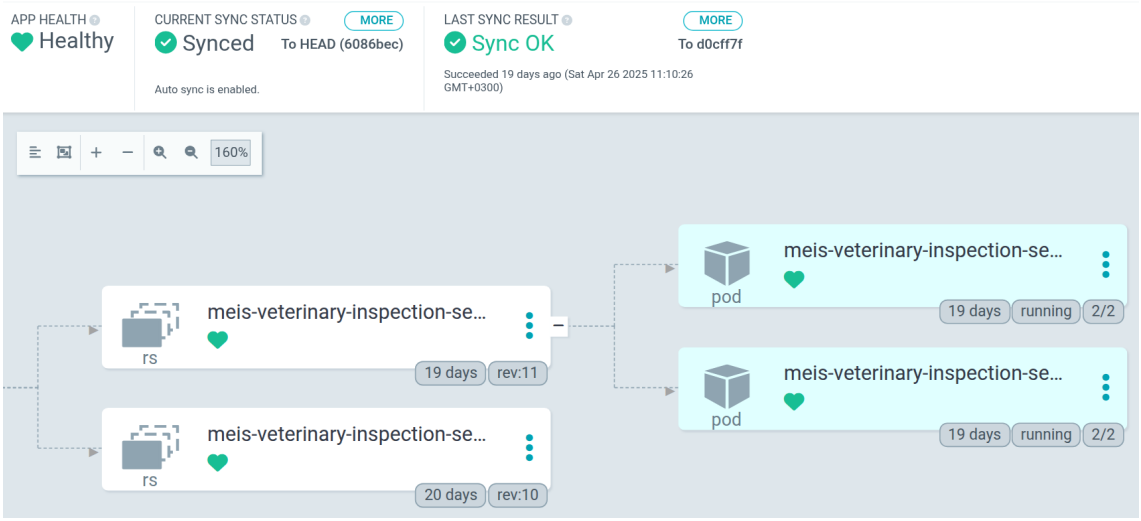
5.2 Kubernetese põhine orkestreerimine

Mikroteenuste juurutamine ja haldamine on hetkel üles ehitatud Kubernetese klastrile, mis tagab skaleeritava, töökindla ja automatiseeritud keskkonna. Selleks, et infrastruktuuri hallata ja rakenduste paigaldus oleks sujuv ning hallatav, on kasutusel Terraform, Helm ja Argo CD, mis koos moodustavad tervikliku lahenduse. [53]

Antud lõputöö raames lisati süsteemi ka uus mikroteenus, mis juurutati olemasolevate kõrvale. Eelduseks oli, et teenus oleks usaldusväärne ja kergesti hallatav, mistõttu pöörati tähelepanu selle kättesaadavusele ja ressursside kasutamise optimeerimisele. Juurutamisel seadistati *replica count* väärtuseks 2, tagamaks, et igal ajahetkel oleks jooksvalt vähemalt kaks aktiivset podi. Lisaks määrati *imagePullPolicy* väärtuseks *IfNotPresent*, et konteineripilti tõmmataks ainult siis, kui see ei ole juba sõlmes olemas. See aitab ära hoida liigset võrgukoormust ning ka tagada rakenduse töö ajal, mil register ei pruugi töötada.

Mikroteenuste paigaldamiseks ja versioonihalduseks kasutatakse Helmi, mis võimaldab luua ja hallata rakenduse juurutuspakette (*Helm Charts*). See lihtsustab rakenduste versioonide haldamist ja tagab keskkonnapõhise seadistamise, võimaldades test-, arendus- ja toodanukeskkondade paindlikku haldamist. Juurutamise automatiseerimiseks on kasutusel

Argo CD, mis võimaldab pidevat juurutamist. See tähendab, et kõik juurutustega seotud muudatused hallatakse läbi koodi repositooriumi, kus Argo CD jälgib muudatusi ning viib need automaatselt Kubernetese klastrisse, kui uus versioon on saadaval. See tagab, et keskkonna seis vastab alati versioonihalduses olevale konfiguratsioonile ning võimaldab muudatusi kiirelt tagasi kerida, kui midagi peaks valesti minema. Samuti on hea visuaalne liides, mille abiga saab mikroteenustel silma peal hoida, mida on näha ka jooniselt 5. [54]



Joonis 5. *Mikroteenuse vaade läbi Argo CD*

6 Tulemuste analüüs

Käesoleva lõputöö eesmärgiks oli luua minimaalselt töötav veterinaarkontrollide mikroteenus koos esmase eesrakendusega, mis toetaks PTA ametnike igapäevast töövoogu loomade kontrollimisel ja andmete haldamisel.

6.1 Funktsionaalsuse võrdlus

Töös keskenduti MVP skoobile, mille eesmärk oli katta veterinaarkontrollide jaoks vajalikud põhitegevused ning pakkuda seejuures täiustusi võrreldes varasema lahendusega. Alljärgnev loetelu kirjeldab arendatud funktsionaalsusi koos märgendusega, kas vastav võimalus eksisteeris ka eelmises süsteemis ning millisel kujul see realiseeritud oli:

- Veterinaarkontrollide loomine, muutmine, kinnitamine ja tühistamine (esines mõlemas, kuid uues süsteemis paremini privileegidega mugavdatud)
- Tegevuskohtade esindusõiguste haldamine (ainult uues süsteemis)
- Tegevuskoha vaate loomine, kus kuvatakse tehtud kontrollid ja prioriteetsed tegevused (ainult uues süsteemis)
- Veterinaarkontrollide üldotsing (esines mõlemas)
- Veterinaarkontrollide raporti funktsionaalsus koos Exceli eksportimise võimalusega (ainult uues süsteemis)
- Failide üleslaadimine ja haldamine (ainult uues süsteemis)
- Avalehe teavitused kinnitamist vajavate kontrollide kohta (esines mõlemas)
- Rollipõhise ligipääsu rakendamine vastavalt kasutajate privileegidele (ainult uues süsteemis)
- Andmete valideerimine ja vigade ennetamine (esines mõlemas, kuid uues süsteemis täiustatud kujul)
- Automaatne logimine ja auditite haldamine (esines mõlemas)
- Mikroteenuse monitoorimine ja mõõdikute kogumine Prometheuse ja Actuatori kaudu (ainult uues süsteemis)

Töö tulemusel valmis lahendus, mis on varasemast oluliselt võimekam ning vastab paremini lõppkasutajate ootustele ja tööprotsessi nõudmistele.

6.2 Lõpptulemused ja arenguprotsess

Autori ülesanne oli arendada välja veterinaarkontrollide mikroteenus, tuginedes kaasaegsele reaktiivsele arendusmudelile. Kuigi autoril oli varasem kogemus klassikalise MVC arhitektuuriga seotud arendustöödes, oli reaktiivne programmeerimine tema jaoks täiesti uus valdkond. Reaktiivse arenduse mõistmine ja rakendamine nõudis alguses märkimisväärset aega ja pühendumist, ent tulemuseks oli edukalt toimiv lahendus ning uue väärtusliku kompetentsi omandamine, mida saab tulevikus rakendada ka muudes projektides ja arendusülesannetes.

Lisaks arendusele kuulus autori vastutusalasse loodud mikroteenuse juurutamine Kubernetese klastritesse. Sellega kaasnes nii automaatsete konveierite seadistamine kui ka keskkondadevaheline testimine ja ressursside optimeerimine, et tagada süsteemi töökindlus ja skaleeritavus.

Arendustöö algfaasis koostas autor koostöös analüütikuga kiirelt esmase Figma prototüübi, mis esitati varajaseks ülevaatuseks süsteemi peamistele kasutajatele ja sidusrühmadele. Prototüüp võimaldas visualiseerida rakenduse töövooge ja kasutajaliidest. Kuigi algne versioon aitas luua üldise ettekujutuse loodava süsteemi funktsionaalsusest, ilmnes esitlemise käigus mitmeid kitsaskohti ja täiendusvajadusi, mida autor algselt polnud ette näinud. Kasutajate kaasamine varases faasis võimaldas neid probleeme kiiresti tuvastada ning suunata süsteemi arendust rohkem tegelikele vajadustele vastavaks. Lisaks prototüübi koostamisele kaasati analüütik ka testimiseks, kes aitas autoril loodud lahendust koos arenduskeskkonnas testida. See võimaldas varakult leida ja parandada loogikavigu ning parandada kasutajamugavust.

Olulise tähelepanu all oli ka failihalduse lahenduse põhjalik läbimõtlemine. Arenduse käigus ilmnes, et loodava süsteemi kasutajate poolt üleslaetavate failide maht osutus suuremaks, kui autor esialgu oli prognoosinud. See sundis failihalduse strateegiat korduvalt ümber hindama ja täiustama. Arutluse all olid mitmed failimahu vähendamise võimalused, sealhulgas erinevad tihendustehnoloogiad. Lahendust valides tuli arvestada, et failide kvaliteet ning failides sisalduvad metaandmed (näiteks pildi loomise aeg, seademeinfo ja muud olulised andmed) säiliksid muutumatuna. Testimiste käigus selgus, et paljud tihendusmeetodid kahjustaksid kas pildi kvaliteeti või eemaldaksid metaandmed.

Arenduse käigus tuli ka kasutajavoogusid kohandada ja täiustada vastavalt praktilistele töökorraldustele ja kasutajate privileegidele. Näiteks tehti muudatus, et veterinaarametnikud saaksid enda loodud kontrolli koheselt kinnitada ilma, et nad peaksid seda eraldi endale suunama, säästes seeläbi aega ja vähendades tarbetut halduskoormust. Samas säili-

tati ka võimalus, et enamiku kontrollide esmast sisestamist teevad veterinaaride abilised, kelle sisestatud andmed vajavad seejärel ametliku kinnituse saamiseks eraldi läbivaatust ja valideerimist.

Tööprotsessi jooksul ilmnes, et paindlikkus ja tihe koostöö kasutajatega olid võtmetähtsusega, et valmiv lahendus vastaks reaalselt väljakujunenud tööprotsessidele ja parandaks ametnike igapäevast töö efektiivsust.

6.3 Tulemuste valideerimine

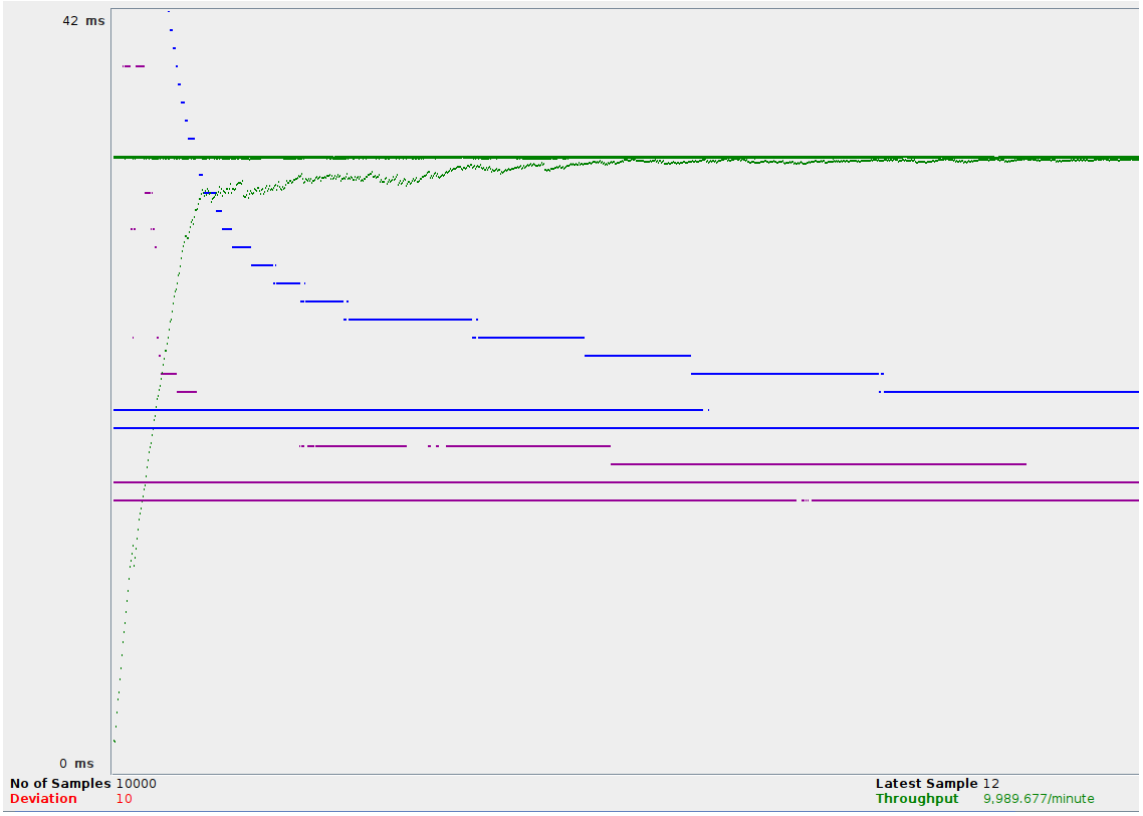
Töö käigus oli üheks oluliseks fookuseks loodud mikroteenuse ja eesrakenduse põhjalik valideerimine, et tagada süsteemi töökindlus, korrektsus ja kasutusmugavus. Autor kasutas valideerimisprotsessis mitmekesist lähenemist, kombineerides automaatset testimist, integreeritud testimist, manuaalset testimist ja kasutajatelt saadud tagasisidet.

Automaatsete testide osas valmis töö jooksul üle 100 testi, mis hõlmasid nii üksusteste kui ka integratsiooniteste. Automaatsete testide kirjutamine võimaldas autoril kiiresti ja turvaliselt muudatusi rakendada, kuna iga muudatus oli koheselt kontrollitav olemasolevate testide abil. See vähendas oluliselt manuaalse testimise koormust ja andis usaldusväärse tagasiside rakenduse tööloogika säilimise kohta. Testide kaudu oli võimalik kiirelt avastada regressioonivead ning vältida varem lahendatud probleemide uuesti esinemist.

Manuaalne testimine oli samuti oluline osa valideerimisprotsessist. Autor kasutas testimiseks Postmani rakendust, mille abil testiti erinevaid API lõpp-punkte, kasutati erinevaid sisendi variatsioone ja kontrolliti vastuste korrektsust. Lisaks viidi läbi manuaalsed töövoogude läbimängimised süsteemi arenduskeskkonnas, mille käigus testiti kogu protsessi algusest lõpuni - alates tegevuskohtade esindusõiguste haldamisest kuni veterinaarkontrollide kinnitamise ja aruandluse eksportimiseni. See võimaldas leida ja kõrvaldada ka selliseid vigu, mis olid tingitud erinevustest lokaalse- ja arenduskeskkonna vahel.

Töö käigus hinnati oluliseks ka süsteemi koormustestimist. Selleks kasutati Apache JMeter tööriista, mille abil viidi läbi koormustestid realistlikes tingimustes. Testis osales 1000 samaaegset kasutajat, kes kõik teostasid valitud põhitoimingud 10 korda järjest. Kasutajate lisandumine oli jaotatud ühtlaselt 60 sekundi jooksul (umbes 16–17 kasutajat sekundis), et simuleerida sujuvat ja järkjärgulist koormuse kasvu. Koormustestide eesmärk oli kontrollida süsteemi stabiilsust ja jõudlust suurenenud kasutajate arvu korral, hinnata andmebaasi reageerimiskiirust ja veenduda, et süsteem suudab koormuse all korrektselt toimida. Testide põhjal selgus, et kõik kontrollitavad päringud on said vastuse piisava

kiirusega ning parendust ei vaja. Alljärgnevalt jooniselt 6 on näha, et süsteem säilitas stabiilse läbilaskevõime ja madalad vastamisajad ka 1000 samaaegse kasutaja koormuse korral, toimides keskmiselt alla 50 millisekundilise vastamisajaga. [55]



Joonis 6. *Veterinaarkontrolli GET-päringu jõudlustesti tulemused JMeteris*

Lisaks tehnilisele testimisele hinnati süsteemi kasutatavust läbi praktiliste ja vabavormiliste intervjuude süsteemi peakasutajatega demode käigus. Nende käigus tutvustati valmis või osaliselt valmis funktsionaalsusi ning koguti kasutajate tagasisidet, et hinnata tööprotsessi sujuvust ja kasutajaliidese selgust. Tagasiside põhjal võib järeldada, et uus süsteem aitab oluliselt vähendada kasutaja ajakulu võrreldes varasema lahendusega – mitmed manuaalsed toimingud on nüüd koondatud ühtsesse töövoogu, vähendades vajadust täiendavate päringute või failide otsimise järele. Kasutajaliidest peeti selgesti mõistetavaks. Tehti küll üksikuid väiksemaid ettepanekuid kasutusmugavuse parandamiseks, kuid üldine hinnang oli väga positiivne. Peakasutajad märkisid, et ootavad süsteemi kasutuselevõttu, kuna see lihtsustab igapäevatööd.

Rakenduse töös esinenud vigade tuvastamiseks rakendati logipõhist kontrolli. Kriitilisemate tõrgete puhul kuvatakse kasutajale koheselt asjakohane veateade. Logidest on samuti võimalik tuvastada andmete terviklikkuse rikkumisi, näiteks välisvõtmete veaolukordi, mis võimaldab väita, et probleemidele saab kiiresti reageerida ja neid analüüsida kõigil logidele ligipääsevatel inimestel.

Kokkuvõttes võib öelda, et süsteemi valideerimine oli mitmekihiline ja põhjalik protsess, mis hõlmas nii tehnilist kui ka kasutajapõhist kinnitust ja aitas tagada, et loodud lahendus vastab nii ärilistele ootustele kui ka tehnilistele nõuetele.

6.4 Tiimi hinnang autorile

Töö lõpufaasis palus autor ka tiimijuhilt ja tiimiliikmetelt, kes olid arenduse ja tööprotsessiga kokku puutunud, anda tagasisidet tehtud töö ja koostöö kohta. Saadud hinnangu põhjal on autor näidanud end väga motiveeritud, professionaalse ja pühendunud tiimiliikmena. Tema lai silmaring, probleemilahendusoskus ning kiire ja konstruktiivne reageerimine erinevatele situatsioonidele avaldasid positiivset mõju kogu tiimi tööõhkkonnale ning aitasid kaasa sujuvale koostööle ja projekti kiirele valmimisele.

Autor eristus oma heatahtliku suhtumise ja tasakaaluka huumorimeelega, mis lõi tiimis positiivse ja toetava töökeskkonna. Lisaks oli autor aktiivne aruteludes, kus tema selge ja väljendusrikas suhtlemisoskus aitas kaasa argumenteeritumale ja sihipärasemale diskussioonile. Märkimisväärne oli ka autori soov mitte ainult probleeme esile tuua, vaid alati pakkuda välja ka võimalikke lahendusi, mis peegeldas süsteemset ja arendusele orienteeritud mõtteviisi.

Kuigi autori nooruslik energia ja entusiasm väljendusid aeg-ajalt kerge kärsitusega, nähakse seda pigem kui potentsiaali, mis kogemuste ja praktika kasvades areneb tugevaks ja tasakaalukaks juhtimisoskuseks. Hinnangu kohaselt on autoril kõik eeldused ja vajalikud isikuomadused, et tulevikus juhtida edukalt nii tehnilisi projekte kui ka meeskondi, ühendades tehnilise pädevuse, sihikindluse ja tugeva inimliku suhtlemisoskuse.

Kokkuvõttes kinnitati tagasisides, et autor on andnud olulise panuse käesoleva projekti õnnestumisse ning on olnud oluline positiivse ja konstruktiivse tiimitöö keskkonna kujundamisel.

6.5 Autori hinnang tööle

Antud projekt oli autori jaoks huvitav ja väljakutseid pakkuv kogemus, mis pakkus võimalust nii professionaalseks edasiarenguks kui ka tehniliste oskuste täiendamiseks. Uued tehnoloogiad ja arendusviisid, millega töö käigus kokku puututi, pakkusid piisavalt süvenemist ja õppimist ning nende valdamine tekitas rahulolu ja enesekindlust.

Autorile on alati meeldinud probleemidele lahendusi otsida ning ka selles projektis andis iga

eduka funktsionaalsuse valmimine ja tehnilise takistuse ületamine tugeva sisemise rahulolu. Üksi töötamine õpetas paremat enesedistsipliini ja pani rohkem rõhku dokumenteerimisele ning koodi kvaliteedile, arvestades, et lahendust võivad hiljem kasutada ja edasi arendada ka teised arendajad.

6.6 Jätkuarendused

Kuigi lõputöö raames valmis veterinaarkontrollide mikroteenuse esmane versioon koos eesrakendusega, on edasiseks arendamiseks kavandatud mitmeid olulisi täiustusi ja laiendusi, mis aitaksid veelgi paremini toetada kasutajate vajadusi ja optimeerida süsteemi tööd.

Üheks suuremaks planeeritavaks edasiarenduseks on Kliendiportaali liidestamine veterinaarkontrollide mikroteenusega. Selle eesmärk on võimaldada loomapidajate ja -käitlejate andmete automaatset ja ajakohastatud edastamist, lihtsustades andmevahetust ja vähendades käsitööd. Samuti on arutatud võimalust teha koostööd JAHISE süsteemiga jahilubade ja jahipidamise osas. Siiski on selle projekti puhul esinenud takistusi, kuna JAHISE teenust haldab hetkel erasektor ja riigil puudub otsene võimalus nõuda arenduste tegemist või API liidestuste avamist. Positiivse märgina on Kliimaministeeriumis arutlusel teema tuua jahilubade menetlemine taas avaliku sektori haldusalasse, mis tulevikus võimaldaks lihtsamat ja ametlikumat integratsiooni. [56, 57]

Teise olulise jätkusuuna ja täiendusena on kavandatud failiarhiivimismehhanismi loomine, mis võimaldaks süsteemi failihalduse muuta pikemas perspektiivis kulutõhusamaks ja hallatavaks. Arvestades, et süsteemis salvestatavate failide hulk võib aja jooksul oluliselt kasvada, näiteks fotod, aruanded ja muud suuremahulised dokumendid, tuleb varakult planeerida nende arhiveerimise ja säilitamise strateegia. Esialgse lahendusena on plaanitud arendada mehhanism, mis liigutaks vanemad ja harvemini kasutatavad failid automaatselt vähemaktiivsele salvestuskihile, näiteks lindirobotisse või mõnda teise külmsalvestuse lahendusse. Selline arhiveerimissüsteem võimaldaks vähendada aktiivse salvestusruumi koormust ja optimeerida kettaruumi kulusid, säilitades samal ajal vajaduse korral ligipääsu ajaloolistele andmetele. Arhiveerimisprotsess võiks põhineda ajapõhistel kriteeriumitel (nt faili loomise kuupäev) või metaandmetel (nt viimase kasutamise aeg). Samas tuleb rõhutada, et lõplik lahendus ja tehnilised detailid tuleb veel täpsemalt kooskõlastada ministeeriumi vastavate osapooltega, et leida kõige sobivam lahendus, mis oleks kuluefektiivne, turvaline ja kasutajate jaoks mugav. Failihalduse strateegia korrektne läbimõtlemine on oluline, et vältida tulevikus ressursipuudust või liigseid kulusid infrastruktuuri ülalhoidmisel.

Lisaks failihalduse täiendamisele on üheks oluliseks jätkuarenduse suunaks ka süsteemi monitoorimisvõimekuse laiendamine. Kuigi rakenduse tööseisundit ja jõudlust jälgitakse juba Grafana kaudu, on plaanis integreerida kogutavad mõõdikud ja logid ka Zabbixisse. Zabbix on ministeeriumis juba kasutusel infrastruktuuri ja muude infosüsteemide jälgimisel ning võimaldab tsentraliseeritud, reaalajas toimivat monitooringut. Zabbixi integratsioon võimaldaks saavutada veelgi varajasema probleemide tuvastamise ning tagada süsteemi töökindluse ja kättesaadavuse ka pikemas ajavahemikus. Samuti võimaldaks tsentraliseeritud monitooring kiiremini avastada jõudlusprobleeme, anomaaliaid või katkestusi enne, kui need mõjutaksid lõppkasutajaid.

7 Kokkuvõte

Antud lõputöö eesmärgiks oli arendada uus mikroteenus koos eesrakendusega, mille kaudu muuta PTA ametnike igapäevatööd hõlpsamaks veterinaarkontrollide läbiviimisel ja loomade heaolu jälgimisel. Töö käigus anti põhjalik ülevaade hetkeolukorrast organisatsioonis, tuues esile olemasolevate süsteemide kitsaskohad ning probleemid, mis vajavad lahendamist. Lisaks kaardistati ja sõnastati nõuded uuele loodavale lahendusele ning kirjeldati tulevikuvisiooni, mille suunas süsteemi edasi arendada. Töös arutleti põhjalikult ka valitud tehnoloogiliste lahenduste üle, analüüsides alternatiive ja põhjendades tehtud otsuseid, alates programmeerimiskeelest ja arendusraamistikest kuni andmebaasi- ja failihalduse ning monitooringu lahendusteni. Arendusprotsessi jooksul pöörati tähelepanu nii süsteemi töökindlusele, kasutusmugavusele kui ka üldisele töövoole, kohendades seda pidevalt vastavalt vajadustele.

Lõputöö lõpuosas anti ülevaade tehtud tööst ja viidi läbi tulemuste analüüs ja valideerimine, mille käigus hinnati loodud süsteemi vastavust seatud eesmärkidele ning planeeriti järgmised võimalikud jätkuarendused. Märkimisväärne oli kasutajatelt ja peakasutajatelt saadud tagasiside, mis kinnitas loodud lahenduse kasutajasõbralikkust ja vastavust tegelikele tööprotsessidele.

Valminud mikroteenus koos eesrakendusega tähistab alles esimest versiooni, mille edasiarendused ja täiustamised jätkuvad ka peale käesoleva töö lõppu. Edaspidised arendused keskenduvad nii uute funktsionaalsuste lisamisele kui ka töövoo lihtsustamisele.

Kasutatud kirjandus

[1] *Loomade heaolu ja kaitse: ELi seadused lahtiseletatult*. [Loetud: 14-04-2025]. URL: https://www.europarl.europa.eu/topics/et/article/20200624STO81911/loomade-heaolu-ja-kaitse-eli-seadused-lahtiseletatult.

[2] *Loomade tervise ja heaolu üle teostab Eestis järelevalvet Põllumajandus- ja Toiduamet*. [Loetud: 14-04-2025]. URL: https://pta.agri.ee/uudised/loomade-tervise-ja-heaolu-ule-teostab-eestis-jarelevalvet-pollumajandus-ja-toiduamet.

[3] *Järelevalve infosüsteem*. [Loetud: 14-04-2025]. URL: https://jvis.agri.ee/jvis/login.html.

[4] *Kubernetes*. [Loetud: 14-04-2025]. URL: https://kubernetes.io.

[5] *Figma*. [Loetud: 14-04-2025]. URL: https://www.figma.com/ui-design-tool/.

[6] *Compare IntelliJ IDEA Editions*. [Loetud: 14-04-2025]. URL: https://www.jetbrains.com/products/compare/?product=idea&product=idea-ce.

[7] *Valitsemisala*. [Loetud: 14-04-2025]. URL: https://agri.ee/ministeerium-uudised-kontakt/ministeerium/valitsemisala.

[8] *Kliendiportaal*. [Loetud: 14-04-2025]. URL: https://portaal.agri.ee/.

[9] José Monteiro Fernando Almeida. "The Role of Responsive Design in Web Development". In: *ResearchGate* (2017). [Loetud: 14-04-2025]. URL: https://www.researchgate.net/profile/Fernando-Almeida-10/publication/324131848_The_role_of_responsive_design_in_web_development/links/5aca790ea6fdcc8bfc84eea8/The-role-of-responsive-design-in-web-development.pdf.

[10] *Veera disainisüsteem 1.0.0*. [Loetud: 14-04-2025]. URL: https://veera.eesti.ee/08be8a71e/p/48af80-veera-disainisusteem-100/.

[11] Aman Malhotra. *9 Best Reasons to Choose Java for Web Development*. Loetud: 14-04-2025. 2019. URL: https://www.xicom.biz/blog/9-best-reasons-to-choose-java-for-web-development/.

[12] *Why Spring?* [Loetud: 14-04-2025]. URL: https://spring.io/why-spring.

[13] *Spring Data R2DBC*. [Loetud: 14-04-2025]. URL: https://spring.io/projects/spring-data-r2dbc.

[14] *WebFlux Security*. [Loetud: 14-04-2025]. URL: https://docs.spring.io/spring-security/reference/reactive/configuration/webflux.html.

[15] *Spring Boot*. [Loetud: 14-04-2025]. URL: https://spring.io/projects/spring-boot.

[16] *Gradle and Maven Comparison*. [Loetud: 14-04-2025]. URL: https://gradle.org/maven-and-gradle/.

[17] *Gradle Vs Maven: What's The Difference?* [Loetud: 14-04-2025]. URL: https://www.interviewbit.com/blog/gradle-vs-maven/.

[18] *PostgreSQL: The World's Most Advanced Open Source Relational Database*. [Loetud: 14-04-2025]. URL: https://www.postgresql.org/.

[19] *Flyway (Redgate) vs. Liquibase in 2025*. [Loetud: 14-04-2025]. URL: https://www.bytebase.com/blog/flyway-vs-liquibase/.

[20] *Database Initialization*. [Loetud: 14-04-2025]. URL: https://docs.spring.io/spring-boot/how-to/data-initialization.html#howto.data-initialization.

[21] *Apache POI™ - the Java API for Microsoft Documents*. [Loetud: 14-04-2025]. URL: https://poi.apache.org/.

[22] *File formats that are supported in Excel*. [Loetud: 14-04-2025]. URL: https://support.microsoft.com/en-us/office/file-formats-that-are-supported-in-excel-0943ff2c-6014-4e8d-aaea-b83d51d46247.

[23] *Java Excel API*. [Loetud: 14-04-2025]. URL: https://jexcelapi.sourceforge.net/.

[24] Oracle. *JDK 21 Documentation*. [Loetud: 14-04-2025]. URL: https://docs.oracle.com/en/java/javase/21/index.html.

[25] *Oracle Java SE Support Roadmap*. Loetud: 14-04-2025. URL: https://www.oracle.com/java/technologies/java-se-support-roadmap.html.

[26] *Spring WebFlux*. [Loetud: 14-04-2025]. URL: https://docs.spring.io/spring-framework/reference/web/webflux.html.

[27] Alexandru-Valentin Catrina. “A Comparative Analysis of Spring MVC and Spring WebFlux in Modern Web Development”. [Loetud: 14-04-2025]. MA thesis. Åland University of Applied Sciences, 2023. URL: https://www.theseus.fi/bitstream/handle/10024/812448/Catrina_Alexandru.pdf?sequence=2.

[28] *Errorprone*. [Loetud: 14-04-2025]. URL: https://errorprone.info/index.

[29] Thomas Broyer. *gradle-nullaway-plugin*. [Loetud: 14-04-2025]. URL: https://github.com/tbroyer/gradle-nullaway-plugin.

[30] *Spotless*. [Loetud: 14-04-2025]. URL: https://dev.to/ankityadav33/standardize-code-formatting-with-spotless-2bdh.

[31] Simon Scholz Lars Vogel. *Introduction to Checkstyle for checking Java code quality - Tutorial*. [Loetud: 14-04-2025]. 2021. URL: https://www.vogella.com/tutorials/Checkstyle/article.html.

[32] *Project Lombok*. [Loetud: 14-04-2025]. URL: https://projectlombok.org/.

[33] *springdoc-openapi v2.8.6*. [Loetud: 14-04-2025]. URL: https://springdoc.org/.

[34] *API Development for Everyone*. [Loetud: 14-04-2025]. URL: https://swagger.io/.

[35] *Jackson vs Gson: Edge Cases in JSON Parsing for Java Apps*. [Loetud: 14-04-2025]. URL: https://dzone.com/articles/jackson-vs-gson-edge-cases-json-parsing-java.

[36] *RabbitMQ*. [Loetud: 14-04-2025]. URL: https://www.rabbitmq.com/.

[37] *Apache Kafka*. [Loetud: 14-04-2025]. URL: https://kafka.apache.org/.

[38] *Jakarta Validation*. [Loetud: 14-04-2025]. URL: https://beanvalidation.org/.

[39] *Vendor-neutral application observability facade*. [Loetud: 14-04-2025]. URL: https://micrometer.io/.

[40] *Prometheus vs Grafana: The Key Differences to Know*. [Loetud: 14-04-2025]. URL: https://betterstack.com/community/comparisons/prometheus-vs-grafana/.

[41] *Explore the potential of Zabbix 7.2*. [Loetud: 14-04-2025]. URL: https://www.zabbix.com/.

[42] *The 5th major version of the programmer-friendly testing framework for Java and the JVM*. [Loetud: 14-04-2025]. URL: https://junit.org/junit5/.

[43] *Mockito*. [Loetud: 14-04-2025]. URL: https://site.mockito.org/.

[44] *MockWebServer*. [Loetud: 14-04-2025]. URL: https://github.com/square/okhttp/tree/master/mockwebserver/.

[45] *Postman*. [Loetud: 14-04-2025]. URL: https://www.postman.com/.

[46] *Testcontainers*. [Loetud: 14-04-2025]. URL: https://testcontainers.com/.

[47] *Testing*. [Loetud: 14-04-2025]. URL: https://projectreactor.io/docs/core/release/reference/testing.html/.

[48] *OAuth and JWT: How To Use Together + Best Practices*. [Loetud: 14-04-2025]. URL: https://workos.com/blog/oauth-and-jwt-how-to-use-and-best-practices/.

[49] *Understanding CORS and CSRF in Spring Boot*. [Loetud: 14-04-2025]. URL: https://medium.com/@vino7tech/understanding-cors-and-csrf-in-spring-boot-7a2d92259592/.

[50] Chip Turner Yann Collet. *Smaller and faster data compression with Zstandard*. [Loetud: 14-04-2025]. URL: https://engineering.fb.com/2016/08/31/core-infra/smaller-and-faster-data-compression-with-zstandard/.

[51] *Bootstrap*. [Loetud: 14-04-2025]. URL: https://getbootstrap.com/.

[52] *Bamboo*. [Loetud: 14-04-2025]. URL: https://www.atlassian.com/software/bamboo/.

[53] *Using ArgoCD & Terraform to Manage Kubernetes Cluster*. [Loetud: 14-04-2025]. URL: https://spacelift.io/blog/argocd-terraform/.

[54] *Argo CD*. [Loetud: 14-04-2025]. URL: https://argo-cd.readthedocs.io/en/stable/.

[55] *Apache JMeter™*. [Loetud: 14-04-2025]. URL: https://jmeter.apache.org/.

[56] *Jahilubade Infosüsteem*. [Loetud: 14-04-2025]. URL: https://jahis.ejs.ee/.

[57] *Kliimaministeerium*. [Loetud: 14-04-2025]. URL: https://kliimaministeerium.ee/.

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

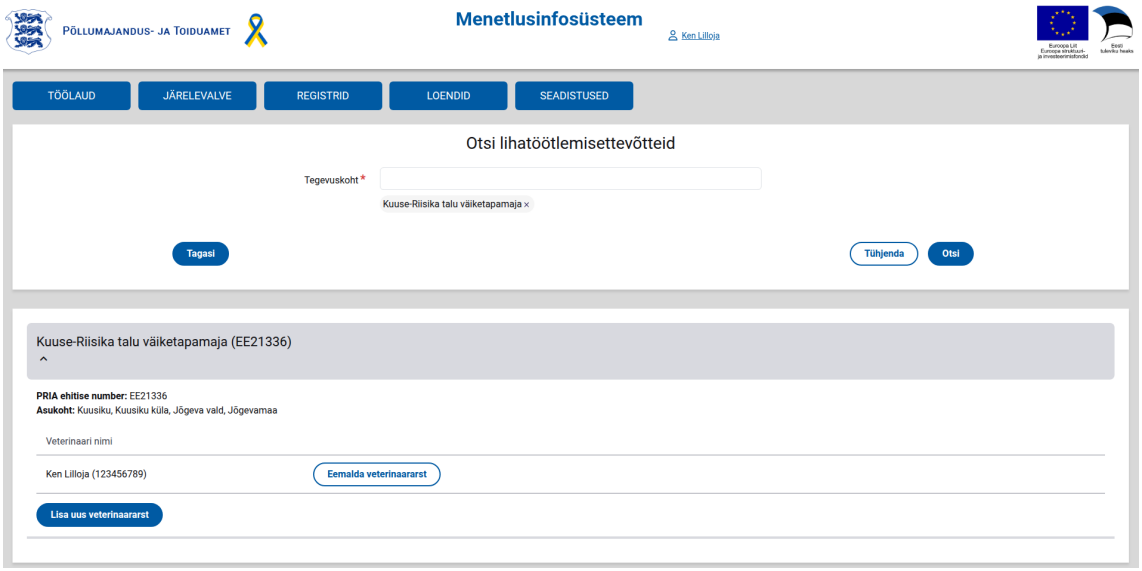
Mina, Ken Lilloja

1. annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Veterinaarkontrollide mikroteenus koos eesrakendusega", mille juhendajaks on Gert Kanter
   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.

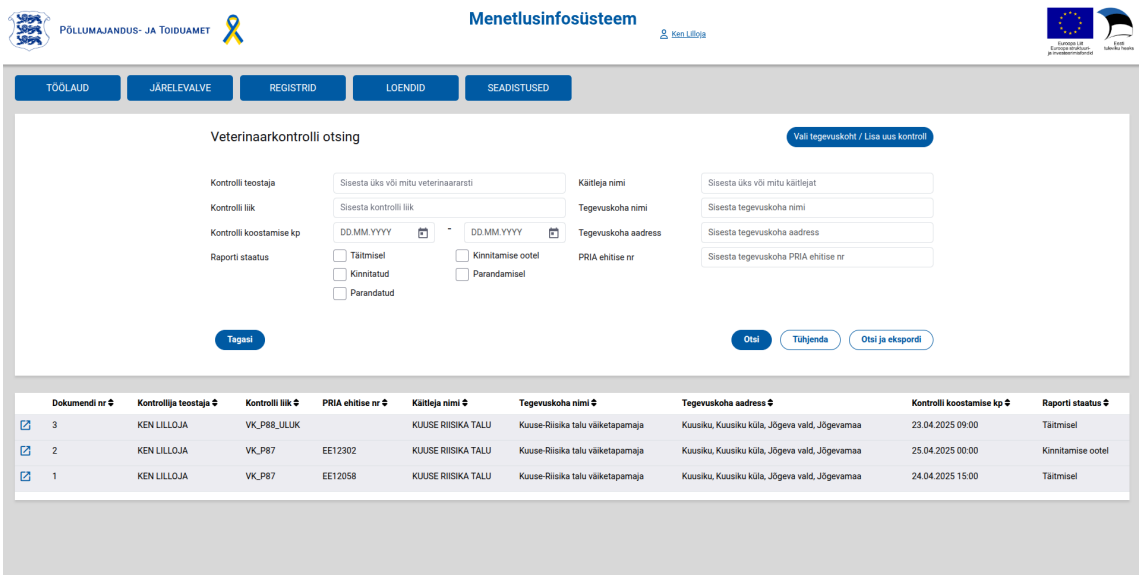
04.06.2025

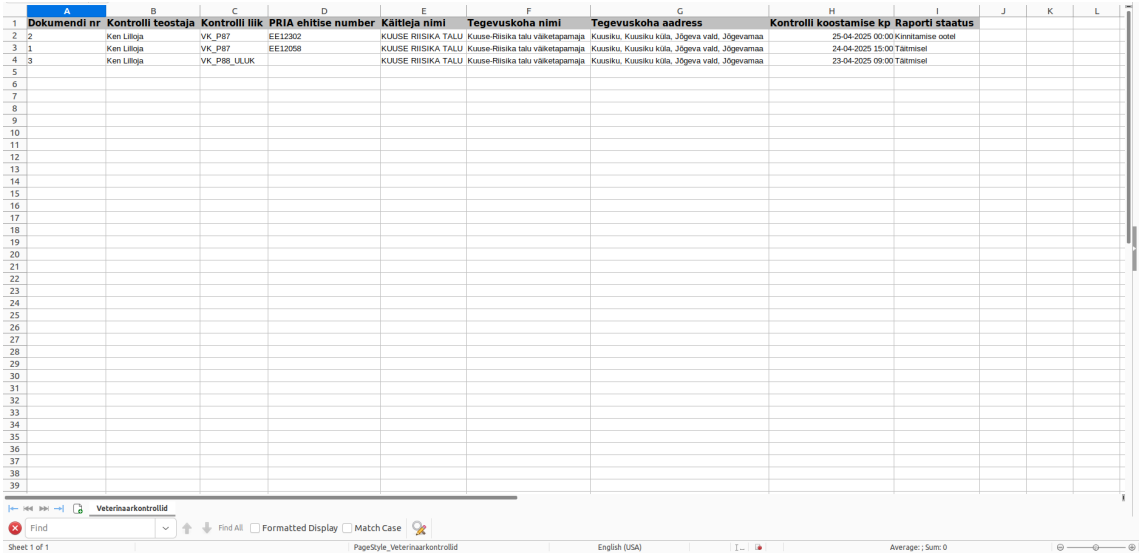
[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või 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.

Lisa 2 - Tegevuskoha esindusõiguse lisamise vaade



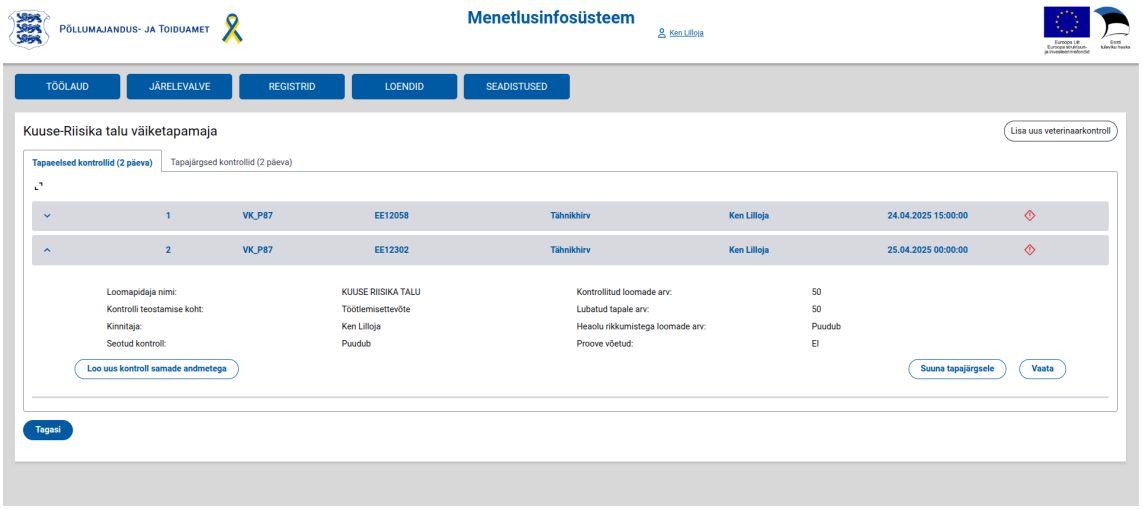
Lisa 3 - Veterinaarkontrollide üldotsing ning eksport Excelisse



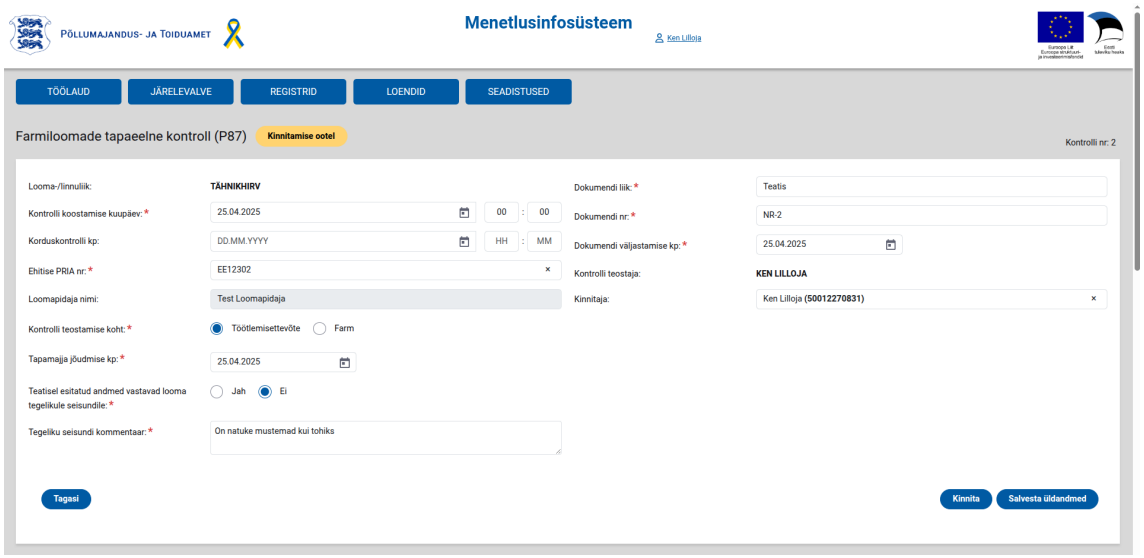


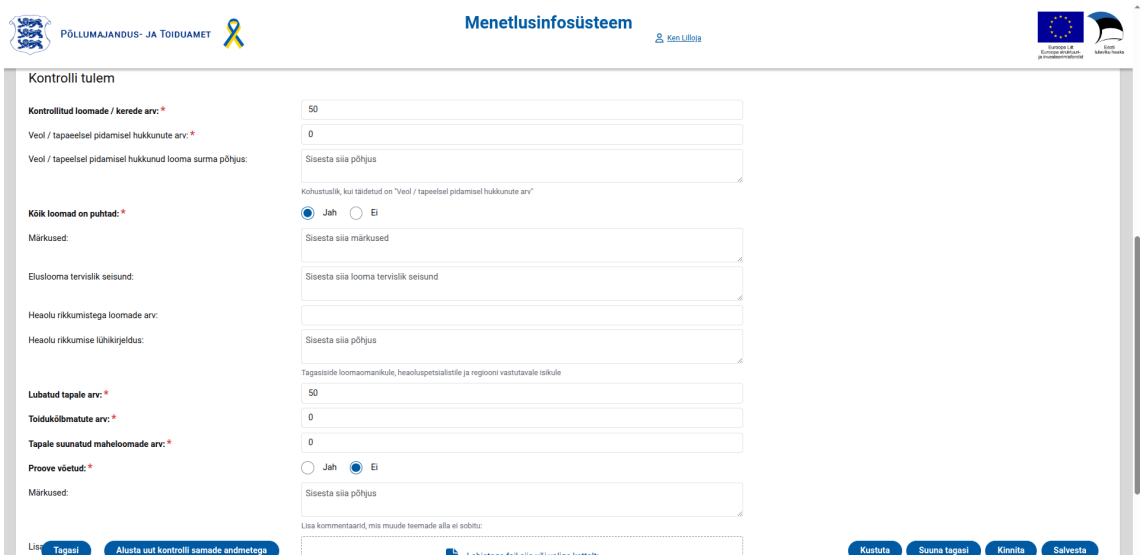
Dokumendi nr	Kontrolli teostaja	Kontrolli liik	PRIA ehitise number	Käitleja nimi	Tegevuskoha nimi	Tegevuskoha aadress	Kontrolli koostamise kp	Raporti staatus
2	Ken Lilloja	VK_P87	EE12302	KUUSE RIISIKA TALU	Kuuse-Riisika talu väiketapamaja	Kuusiku, Kuusiku küla, Jõgeva vald, Jõgevamaa	25-04-2025 00:00	Kinnitamise ootel
1	Ken Lilloja	VK_P87	EE12058	KUUSE RIISIKA TALU	Kuuse-Riisika talu väiketapamaja	Kuusiku, Kuusiku küla, Jõgeva vald, Jõgevamaa	24-04-2025 15:00	Täitmisel
3	Ken Lilloja	VK_P88_ULUK		KUUSE RIISIKA TALU	Kuuse-Riisika talu väiketapamaja	Kuusiku, Kuusiku küla, Jõgeva vald, Jõgevamaa	23-04-2025 09:00	Täitmisel

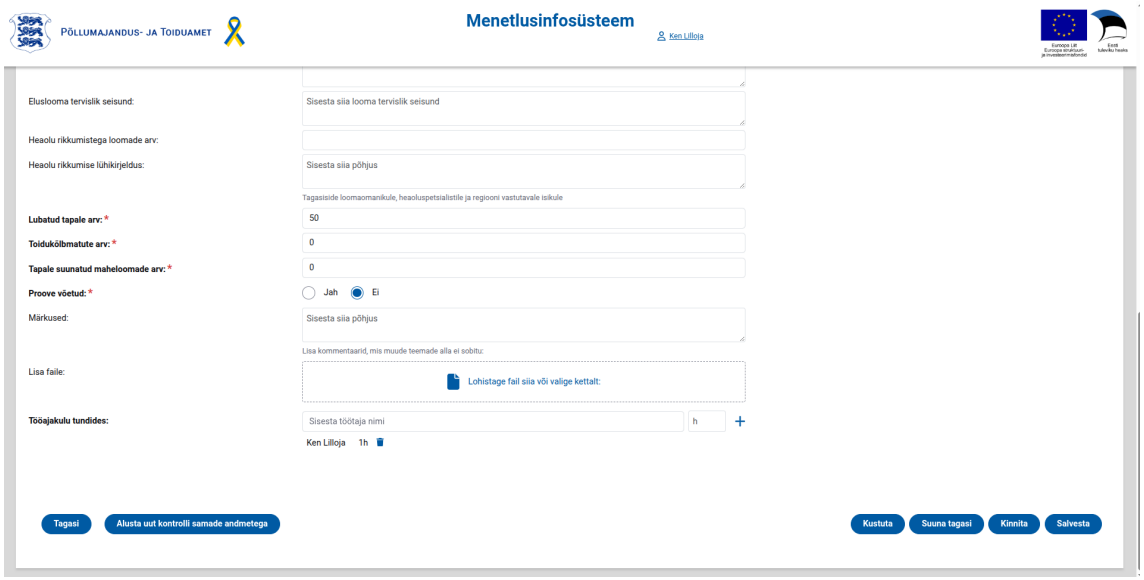
Lisa 4 - Tegevuskoha sisene vaade kontrollidele



Lisa 5 - Veterinaarkontrollide muutmise vaade






PÕLLUMAJANDUS- JA TOIDUAMET
Menetlusinfosüsteem
Ken Lilloja
Elusloomа tervislik seisund:
Sisesta siia looma tervislik seisund
Heaolu rikkumistega loomade arv:
Heaolu rikkumise lühikirjeldus:
Sisesta siia põhjus
Tagasiside loomaomanikule, heaoluspetsialistile ja regiooni vastutavale isikule
Lubatud tapale arv: *
50
Toidukõlbmatute arv: *
0
Tapale suunatud maheloomade arv: *
0
Proove võetud: *
Jah
Ei
Märkused:
Sisesta siia põhjus
Lisa kommentaarid, mis muude teemade alla ei sobitu:
Lisa faile:
Lohistage fail siia või valige kettalt:
Tööajakulu tundides:
Sisesta töötaja nimi
h
Ken Lilloja 1h
Tagasi
Alusta uut kontrolli samade andmetega
Kustuta
Suuna tagasi
Kinnita
Salvesta

Lisa 6 - Töölaua teavituse vaade