TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Priit Rätsep 176036IDDR

Päästeameti ohutuse infosüsteemi tulekahjujuhtumite haldusmooduli arendus

Diplomitöö

Juhendaja: Meelis Antoi
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: Priit Rätsep

11.04.2021

Annotatsioon

Käesoleva diplomitöö eesmärgiks oli arendada tulekahjujuhtumite haldusmoodul Päästeameti ohutuse infosüsteemile. Loodav rakendus peab vastama Päästeameti uutele tulekahjujuhtumite haldamise nõuetele. Tagarakendus peab olema arendatud põhirakendusest eraldi seisvana ning kasutajaliides peab sarnanema põhirakendusele.

Diplomitöös kirjeldatakse varasemalt kasutuses olnud lahendust ning uue lahenduse vajaduse loonud probleemi olemust. Analüüsi osa keskendub loodava rakenduse nõuetele ning klientrakenduse ja tagarakenduse arenduseks võimalike tehnoloogiate ja komponentide valimisele. Analüüsitakse ka rakenduse arhitektuuri. Diplomitöö teostuse osa katab andmebaasi disainimist ning klientrakenduse ja tagarakenduse arendust. Põgusalt on juttu rakenduse turvalisusest ning lõpuks on kirjeldatud välistest allikatest vastu võetavate tulekahjujuhtumite testimist. Tulemuseks on töötav veebirakendus ohutuse infosüsteemi osana.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 37 leheküljel, 6 peatükki, 25 joonist, 3 tabelit.

Abstract

Development of Fire Incident Management Module for Estonian Rescue Board Safety Information System

The aim of this thesis was to develop a fire incident management module for Estonian Rescue Board safety information system. The application must meet the new fire incident management requirements of the Rescue Board. Backend must be developed separately from the main application and user interface must be similar to the main application.

The thesis describes the previously used solution and the nature of the problem that created the need for a new solution. The analysis part focuses on the requirements of the application, the selection of possible technologies and components for the development of client and backend applications, and analysis of the application architecture. The implementation part of the thesis covers the design of the database and the development of client and backend applications. Security of the application is also briefly discussed and finally the testing of fire incidents from external sources is described. The end result is a working web application as a part of safety information system.

The thesis is in Estonian and contains 37 pages of text, 6 chapters, 25 figures, 3 tables.

Lühendite ja mõistete sõnastik

AMQP	*Advanced Message Queuing Protocol*, avatud standardprotokoll sõnumivahetuseks erinevatel platvormidel [1]
API	*Application Programming Interface*, liides rakenduste vaheliseks suhtluseks
CLI	*Command Line Interface*, käsurea liides
DOM	*Document Object Model*, dokumendi objektimudel
GET	HTTP meetod andmete pärimiseks
GUI	*Graphical User Interface*, graafiline kasutajaliides
HTML	*HyperText Markup Language*, standardne veebilehtede märgendkeel
HTTP	*Hypertext Transfer Protocol*, kliendi ja serveri vahelise suhtluse protokoll [2]
JMS	*Java Message Service*, sõnumite loomise, saatmise, vastu võtmise ja lugemise standard Java EE baasil rakenduste komponentidele [3]
JPA	*Java Persistance Api*, Jakarta EE API spetsifikatsioon relatsiooniliste andmete ja Java objektide seostamiseks
JSONB	*JSON Binary*, binaarsele kujule viidud JSON failiformaat
JVM	*Java Virtual Machine*, Java virtuaalmasin
NPM	*Node Package Manager*, JavaScript-i teekide haldamise tööriist
OIS	Ohutuse infosüsteem
PDF	*Portable Document Format*, digitaalsete andmete loomise, haldamise, kogumise ja edastamise formaat [4]
POST	HTTP meetod andmete sihtkohta saatmiseks
PPA	Politsei- ja Piirivalve amet
PÄVIS	Pääste sündmuste, teenistusraamatu ja ressursside halduse infosüsteem

REST	*Representational State Transfer*, arhitektuuriline stiil hajussüsteemide suhtluseks veebis [5]
SMTP	*Simple Mail Transfer Protokol*, standardprotokoll elektrooniliste kirjade saatmiseks ja vastu võtmiseks
SOS	Hädaabiteadete menetlemise andmekogu
SPA	*Single-Page Application*, üheleherakendus
SQL	*Structured Query Language*, andmebaasiga suhtlemise päringukeel
UAA	Cloud Foundry kasutajakonto ja autentimise teenus [6]
XML	*Extensible Markup Language*, lihtne teksti baasil märgendkeel struktuurse info esitamiseks [7]
ZIP	Faili kokku pakkimise formaat

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

OIS (Ohutuse infosüsteem) on Päästeameti poolt kasutatav infosüsteem, mille põhieesmärgiks on Päästeameti andmete kogumine, töötlemine ning menetluste ja toimingute haldamine tagamaks päästeasutuse ülesannete kiire ja efektiivse täitmise ja tõhusa järelevalve korralduse [8]. Lisaks paljule muule kogub ja haldab OIS tulekahju pääste-juhtumite andmeid. Tulekahjujuhtumite andmeid kogutakse OIS-i kahest teisest infosüsteemist, milleks on SOS (Hädaabiteadete menetlemise andmekogu) ning PÄVIS (Pääste sündmuste, teenistusraamatu ja ressursside halduse infosüsteem). PÄVIS tegeleb tulekahjujuhtumite vaates peamiselt sündmuskoha andmete haldamisega. Edasi menetlevad OIS-s tulekahjujuhtumeid Päästeameti menetlejad.

Arendamisel on PÄVIS uus versioon PÄVIS2, milles on Päästeameti poolt üle vaadatud ja muutunud vajaduste põhjal uuendatud tulekahjujuhtumite haldamine. Sellega kaasneb ka OIS-i saadetavate andmete muudatus ja vajadus uue tulekahjujuhtumite mooduli järele.

OIS on aja jooksul kasvanud väga suureks ja siiani on seda arendatud ühe suure monoliitrakendusena. Ettevõtte uued arendused liiguvad aga mikroteenuste ja teenusepõhiste arhitektuuride suunal ning sellest peab ka OIS kinni.

Käesoleva diplomitöö eesmärgiks on luua Ohutuse infosüsteemi juhtumite halduse moodul, mis võimaldaks SOS-i ja PÄVIS2 andmeid vastu võtta ja töödelda ning Päästeameti menetlejatel hallata olemasolevaid ja vajadusel luua uusi tulekahjujuhtumeid. Lähtudes ettevõtte arendusjuhistest peab loodav rakendus olema põhirakendusest eraldiseisev. Kuna olemasoleva tulekahjujuhtumite lahenduse migreerimine eraldiseisvaks mooduliks oleks piisavalt keerukas ning uued Päästeameti poolt esitatud vajadused on olemasolevast väga erinevad, siis otsustas töö autor rakenduse nullist luua.

Rakendus lihtsustab Päästeameti menetlejate tööd võimaldades tulekahjujuhtumite lõplikku menetlemist ning annab ka politseile põhjaliku ülevaate juhtumitest, kui on vaja

andmeid leida ja kontrollida. Kogutud andmed võimaldavad tõhusamat ennetustegevust ja ohutusjärelevalvet. Tänu statistikale ja tulekahju tekkepõhjuste väljavõtete analüüsile saab asutus sisendi juhtimisotsuste tegemiseks ja strateegiliste eesmärkide püstitamiseks.

2 Metoodika

Käesoleva diplomitöö eesmärgiks on arendada Päästeameti ohutuse infosüsteemile välja uus tulekahjujuhtumite haldamise moodul klientrakenduse ja tagarakenduse näol, mis vastaks uuele Päästeameti poolt välja töötatud andmekomplektile. Uue lahenduse loomisel tuleks arvesse võtta ka vanas lahenduses esinevad probleemid.

Analüüsi osas on kirjeldatud rakendusele esitatud funktsionaalsed ja mittefunktsionaalsed nõuded, võrreldakse erinevaid komponente ja töövahendeid ning valitakse välja nende seast parimad kandidaadid lahenduse realiseerimiseks. Esitatud on ka loodava rakenduse arhitektuuri analüüs.

Teostuse osa kirjeldab töö autori poolt läbitud protsessi, ette tulnud probleeme klientrakenduse ja serveripoolse teenuse arendamisel, andmebaasi disaini valikuid ning rakenduse testimise protsessi.

Lõpuks võetakse kokku tehtud töö tulemus, mis sellega paremaks muutus ja millist majanduslikku efekti võiks oodata. Lisaks antakse ülevaade ka veel ees ootavatest arendustest, mida käesolevas töös ei käsitleta.

3 Olemasolev lahendus

Tulekahju pääste-juhtum on üks SOS ja PÄVIS infosüsteemides esineva pääste juhtumi alaliik. Ohutuse infosüsteemi üheks tähtsaks osaks on nende tulekahjujuhtumite haldamine. Olemasolev lahendus võtab vastu ja töötleb juhtumite andmeid samuti SOS-st ning veel hetkel kasutusel olevast PÄVIS-st. Antud töö kirjutamise hetkel kasutuses olevas lahenduses on tulekahjujuhtumite peamiseks eristamiseks kasutusel sündmuse liigi mõiste. Sündmused on jagatud kuude liiki, millest viimane on omakorda jaotatud üheteistkümneks alaliigiks:

- Ekslik väljakutse
- Hoone tulekahju
- Järelkustutus
- Topeltkutse
- Tulekahju hoones
- Tulekahju väljaspool hoonet

Vastavalt tulekahju liigi valikule on menetlejale täitmiseks vorm küsimustikuga, mis kasutaja vaates on andmete iseloomu järgi jagatud kaheksaks alamsakiks. Peale kohustuslike küsimuste täitmist saab menetleja juhtumi kinnitada, millega loetakse see lõppenuks.

Lihtsustatult ja üldjuhul on põhitöövoog selline, et tulekahjujuhtum saadetakse SOS-st OIS-i ja PÄVIS-sse. Peale PÄVIS-s menetlemist ja kinnitamist saadetakse see omakorda sealt OIS-i, kus toimub juhtumi edasine menetlemine. OIS-s teeb menetleja vajalikud muudatused temale teada oleva info põhjal ning seejärel kinnitab juhtumi.

Päästeameti vajadused tulekahjujuhtumite halduse ja liikide määratluse suhtes on aga aja jooksul muutunud, millest on tingitud ka vajadus uue lahenduse järele. Uut süsteemi luues on ka hea hetk uurida vana lahenduse probleeme, et nendest edaspidi hoiduda. Üheks

suuremaks probleemiks on see, et kogu ohutuse infosüsteem on loodud ühe suure monoliitrakendusena, mistõttu on arendajal seda keeruline hallata. Ka automaattestidega kaetus võiks olla parem, mis aga projekti mahukuse tõttu antud töö skoopi ei mahu.

Praeguses lahenduses on andmebaasi disainimisel kasutatud mitmeid mahukaid tabeleid, mis võib tekitada jõudlusprobleeme ja teeb rakenduses logitavate SQL päringute jälgimise väga keeruliseks. Päästeametil on ka tihtipeale vaja väljavõtteid, mille jaoks pole rakendusse funktsionaalsust sisse ehitatud. Andmebaasi disainimisel peaks arvestama, et päringuid oleks võimalikult lihtne teha ka sellistel kasutajatel, kes igapäevaselt andmebaasidega ei tööta.

On esinenud ka probleeme andmete saatmisel klientrakendusest, mille käigus osa andmeid on läinud kaduma. Praeguses lahenduses saadetakse salvestamisel kogu juhtumi andmestik ühe päringuna, mis võib olenevalt täidetuse astmest osutuda väga mahukaks. See on aga kaasa toonud juhud, kus osa sõnumist on kaotsi läinud.

4 Analüüs

Käesolevas peatükis esitatakse loodava rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded, analüüsitakse rakenduse arhitektuuri, komponentide ja töövahendite ning andmebaasi ja selle disaini valikuid.

Uue lahenduse kasutusele võtuga kaasnevateks suurimateks muudatusteks kasutajale on uuest andmekomplektist tulenevad andmeväljade ja valikute muutused ning tüüpjuhtumi, tüüpjuhtumi täpsustuse ja staatuse mõistete sisse toomine, mis asendavad senist sündmuse liiki ja alaliiki. Tüüpjuhtumi üheks oluliseks erinevuseks võrreldes vana liigitusega on näiteks eri tüüpi sõidukite tulekahjude määratlus. Kasutusele võetakse ka uuemad raamistikud ning arhitektuurilise poole pealt liigutakse monoliitrakenduse pealt teenusepõhise arhitektuuri suunal. Kuna uus teenus ei ühendu enam otse põhirakenduse andmebaasi külge, siis tuleb ka andmebaasi disainis muudatusi teha, et vältida üleliigseid rakenduste vahelisi päringuid.

Töö eesmärgiks on Päästeametile arendada töötav veebirakenduse moodul ohutuse infosüsteemi, mis oleks uutele nõuetele vastav. Sinna alla käib nii klientrakenduse kui ka tagarakenduse loomine. Projekti mahukuse tõttu jääb töö skoobist välja autentimise ja turvalisuse detailne kirjeldus. Samuti ei käsitleta antud töös produktsioonikeskkonna ja pideva integratsiooni lahenduse üles seadmist ja konfigureerimist, mis on tehtud ettevõtte meeskonna süsteemiadministraatori poolt. See osa piirneb probleemi lahenduseks vajaliku protsessi kirjeldusega autori arvutis. Töö hõlmab ka lahenduse testimiseks läbi viidavaid andmete vastuvõtu koormusteste ettevõtte testkeskkonnas.

4.1 Nõuded

Tulenevalt PÄVIS2 uuenenud andmekomplektist ja Päästeameti muutunud vajadustest on koostatud rakenduse nõuded koostöös OIS meeskonna, Päästeameti ja PÄVIS2 meeskonnaga. Nõuete kirjeldamisel on kõigepealt välja toodud sellised, mis kehtivad kõigi kasutajagruppide kohta ning seejärel on lähtutud kasutajagruppide õigus- ja vajaduspõhisusest. Eraldi on kirjeldatud süsteemi nõuded. Silmas tuleb pidada, et

kasutajagrupid koosnevad kasutajatest, kellel on kogu OIS süsteemi vaates siiski määratud kindlad kitsamad rollid. Nõuete kirjeldamisel ei ole eraldi üksikasjalikult välja toodud juhtumi detailvaate kõiki nõudeid.

Tulekahjujuhtumitega tegelemisel võib kasutajad vajaduste ja õiguste järgi jagada kolme gruppi:

- Vaatleja – kasutaja, kes otseselt menetluse protsessis ei osale, aga kellel võib tekkida vajadus juhtumi kohta infot saada.
- Menetleja – tulekahjujuhtumi menetlemise protsessis osalev kasutaja, kellel on õigus teha menetluse käigus juhtumi juures muudatusi.
- Välisvaatleja – piiratud õigustega vaatleja, kes pole igapäevaselt OIS kasutaja, aga kellel võib oma töö käigus vaja minna tulekahjujuhtumite andmeid. Näiteks võib selliseks isikuks olla PPA töötaja.

4.1.1 Funktsionaalsed nõuded

- Kasutaja peab saama otsida kõiki tulekahjujuhtumeid.
- Kasutaja peab saama otsingus filtreerida juhtumeid juhtumi aadressi, regiooni, maakonna, linna, juhtumi numbri, oleku, tekkepõhjuste, ehitise nimetuse, tüüpjuhtumi, tüüpjuhtumi täpsustuse, staatuse, järgnenud menetluse liigi, dokumendi liigi, kinnitaja ja toimumise aja järgi.
- Kasutaja peab saama näha juhtumi detaile.
- Kasutaja peab saama navigeerida juhtumi asukohta Maa-ameti X-GIS kaardil.
- Kasutaja peab saama avada juhtumi PÄVIS2-s juhul kui juhtum on seal kinnitatud ja OIS-i saadetud.
- Kasutaja peab saama avada juhtumi SOS-s juhul kui juhtum on sealt saadetud.

Vaatleja:

- Vaatleja peab saama alla laadida juhtumi detailinfoga PDF dokumendi.

- Vaatleja peab saama alla laadida juhtumi juurde lisatud dokumente.
- Vaatleja peab saama teha filtriga määratud otsingu tulemuste põhjal statistika CSV [9] väljavõtte.

Menetleja:

- Menetleja peab saama käsitsi luua uusi juhtumeid.
- Menetleja peab saama määrata juhtumi liiki tüüpjuhtumite näol, milleks on tulekahju hoones, tulekahju väljaspool hooneid, tulekahju metsas või maastikul, tulekahju alarm (ATeS), maismaasõiduki tulekahju, veesõiduki tulekahju, raudteesõiduki tulekahju, õhusõiduki tulekahju.
- Menetleja peab saama juhtumit kinnitada, kui kõik validatsioonid läbi lähevad.
- Menetleja peab saama juhtumit kustutada, kui tegemist on õppusega või juhtum toimus mõne muu riigi territooriumil.
- Menetleja peab saama juhtumit siduda OIS ehitisega.
- Menetleja peab saama juhtumi andmeid sisestada ja muuta vastavalt valitud tüüpjuhtumi andmestikule.
- Menetleja peab saama alla laadida juhtumi detailinfoga PDF dokumendi.
- Menetleja peab saama juhtumile lisada dokumente.
- Menetleja peab saama alla laadida juhtumi juurde lisatud dokumente.
- Menetleja peab saama teha filtriga määratud otsingu tulemuste põhjal statistika CSV väljavõtte.
- Menetleja peab rakenduse toel saama arvutada juhtumi käigus tekkinud varakahju ja päästetööga ära hoitud varakahju (vastavalt Sisekaitseakadeemia välja töötatud metoodikale).
- Menetleja peab saama edastada juhtumi teisele kasutajale e-kirjas.

- Menetleja peab saama määrata tulekahjujuhtumi aadressi kaardilt (salvestades ads_oid).

Välisvaatleja:

- Välisvaatleja peab enne juhtumi avamist lisama juhtumi vaatamise põhjenduse.
- Välisvaatleja peab saama sama juhtumit avada 24 tunni jooksul ilma uuesti põhjust selgitamata.

Süsteem:

- Süsteem peab kinnitatud tulekahjujuhtumite andmeid vastu võtma SOS-st.
- Süsteem peab tulekahjujuhtumite andmeid vastu võtma PÄVIS2-st.
- Süsteem peab teiste allikate andmed seotama ühe juhtumi kohta käivaks andmekomplektiks.

4.1.2 Mittefunktsionaalsed nõuded

- Süsteem peab töötama ettevõtte poolt ette antud Google Chrome ja FrontMotion Firefox veebilehitsejate versioonidel.
- Kasutajaliides peab olema eesti keelne.
- Lisatud dokumendid tuleb salvestada S3 ühilduvasse andmehoidlasse.
- Tagarakendus peab olema OIS põhirakendusest eraldiseisev.

4.2 Komponendid ja tehnoloogiad

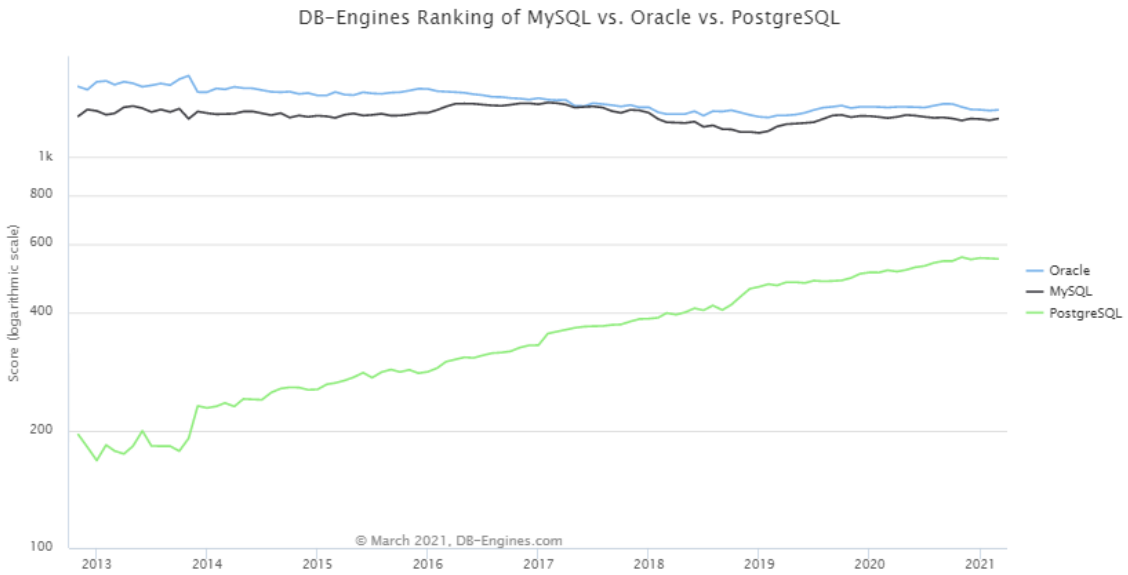
Järgnevas osas analüüsitakse erinevate võimalike tehnoloogiate sobivust antud projekti ja valitakse välja sobivaimad tehnoloogiad.

4.2.1 Andmebaas

OIS üheks suureks osaks on erinevate väljavõtete tegemine aastate jooksul kogunenud andmete kohta. Nende põhjal tehtud statistika aitab välja selgitada kas liigutakse õiges suunas või tuleb teha lähenemises muudatusi. Varasemalt toimus nende kümneid kuni

sadu tuhandeid kirjeid sisaldavate väljavõtete tegemine otse põhirakenduses ja selliste päringute tegemine võis võtta aega mitmeid minuteid. Aruannete esitamise perioodil tegid selliseid päringuid palju inimesi samal ajal, mis oli tohutu koormus ning selle tõttu oli häiritud kogu rakenduse töö. Selle jaoks arendati välja eraldi mikroteenusena aruandluse rakendus, milles on kasutusel PostgreSQL. Sinna replitseeritakse vajalikud tabelid ja andmed kõige rohkem koormust saavatest OIS teenustest. Ka tulekahjujuhtumite statistika järele on pidev vajadus, mille tõttu on vaja ka loodava rakenduse andmebaasi replitseerida aruandluse baasi. Üheks määravaks teguriks andmebaasi valikul saigi asjaolu, et erinevate andmebaaside haldussüsteemide vaheline andmete replitseerimine on keerulisem kui seda on näiteks ühest PostgreSQL andmebaasist teise.

Ettevõttes, millele loodavat rakendust tehakse, on küll kasutusel erinevaid andmebaaside haldussüsteeme, aga kõige enam kasutust on leidnud PostgreSQL. PostgreSQL on pidevas arenduses vabavaraline objekt-orienteeritud relatsiooniline andmebaasi süsteem, mis on tuntud jõudluse ja töökindluse poolest [10]. Joonis 1 kujutab graafikut Oracle, MySQL ja PostgreSQL andmebaaside populaarsusest aja jooksul [11]. Võrreldes teistega, mille populaarsus on aja jooksul võrdlemisi samaks jäänud, on PostgreSQL populaarsus kiires kasvutrendis. See näitab, et kasutajad näevad seal potentsiaali ja võimekust.



Joonis 1. Andmebaaside populaarsuse võrdlus läbi aja

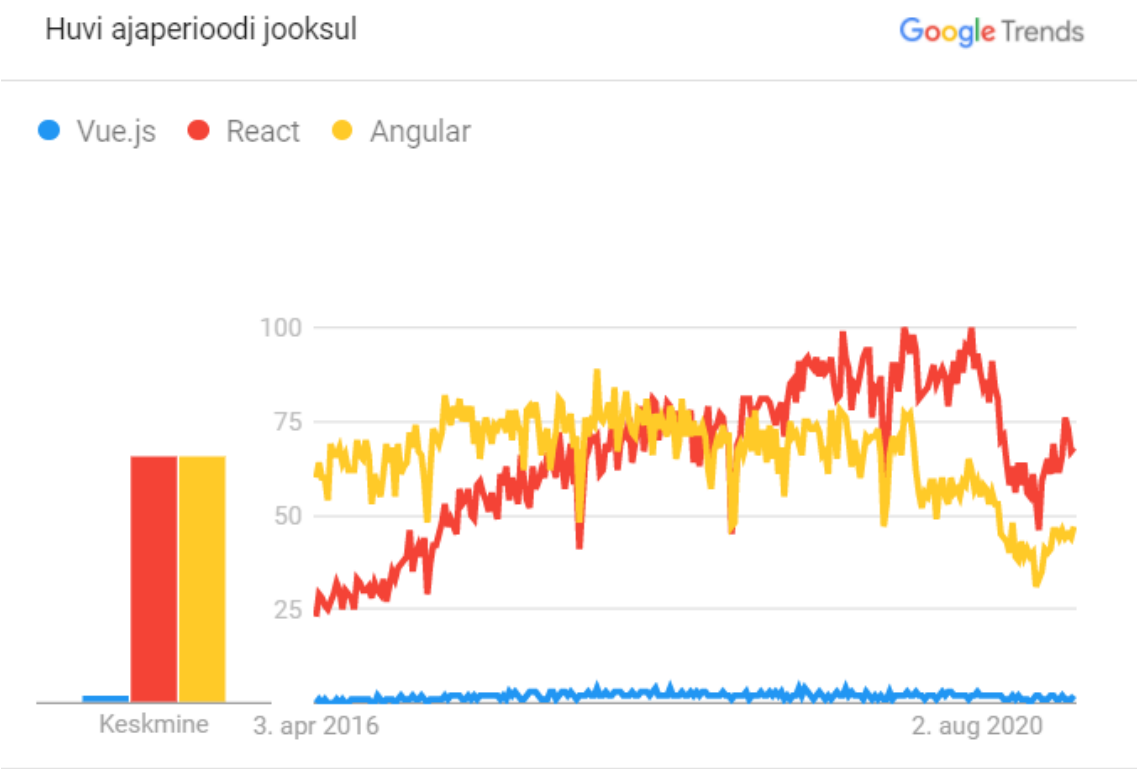
Võttes arvesse andmete replitseerimise vajadust, PostgreSQL pidevat arengut ning asjaolu, et PostgreSQL on relatsiooniliste andmebaaside populaarsuse poolest neljandal kohal [12] ja viie populaarseima andmebaasi hulgast kõige kiiremini populaarsust koguv, valis töö autor loodava rakenduse andmebaasiks PostgreSQL.

4.2.2 Klientrakendus

Frontend ehk kliendipoolse rakenduse arendamiseks on väga palju erinevaid võimalusi nii raamistike näol kui ka arhitektuuri mõttes. OIS põhirakenduses on klientrakendus ehitatud teenuse poolega kokku ja raamistikuna on seal kasutusel AngularJS ehk Angular 1 [13].

AngularJS on JavaScripti baasil dünaamiliste veebirakenduste ehitamiseks mõeldud kõrgema abstraktsioonitaseme raamistik, mille eelisteks HTML ja JavaScripti ees on *data binding* ehk automaatne andmete sünkroniseerimine vaate ja mudeli komponendi vahel, vormide ja vormi valideerimise toetus, DOM elementidele funktsionaalsuse lisamine ja koodi korduvkasutatavateks komponentideks koondamine [14]. Kuna AngularJS aktiivne arendus lõppes aastal 2018 ja pikaajaline tugi lõpeb 2021. aasta lõpus [15], siis on vaja leida uus lahendus.

Ettevõtte poolt on klientrakenduse jaoks heakskiidu saanud Angular(2+) raamistik ning Vue.js ja React JavaScript teegid, milledest töö autoril on kõige rohkem kogemust Angulariga. Kõik kolm on aktiivses arenduses ja saavad pidevalt uusi täiendusi ning kõigil on olemas korralik dokumentatsioon, aga suur osa Vue.js materjalidest on hiinakeelsed [16]. Projekti GitHubi repositooriumit peetakse väga heaks populaarsuse indikaatoriks. Tabel 1 on näha Angular, Vue ja Reacti statistikat, mille järgi on näha, et Angular on teistest mõne võrra maas [17]. Teisest küljest Google Trends viimase viie aasta statistika Joonis 2 näitab, et Vue-d on teisest kahest palju vähem esile tulnud. Tehnoloogia populaarsus võib olla heaks näitajaks kui lihtne on arenduse käigus probleemidele lahendusi leida, kuna dokumentatsioonist võib jääda väheseks ja mida populaarsem on tehnoloogia, seda tõenäolisemalt on keegi teine juba varem sama probleemiga kokku puutunud.



Joonis 2. Angular, React ja Vue Google Trends viimase viie aasta statistika

Tabel 1. Angular, React ja Vue GitHub meetrikad

	Angular	**React**	**Vue.js**
Jälgijaid	3.2k	6.7k	6.3k
Tähti[1]	70.9k	154k	200.8k

Valiku tegemisel on tähtis arvestada ka raamistikuga kaasas olevate põhifeatuuridega, mida loodavas rakenduses vaja võib minna. Nendest kolmest on Angular ainuke, millel on sisse ehitatud HTTP suhtluse, vormide protsessimise ja valideerimise funktsionaalsus [18] ning lisaks on saadaval RxJS [19] teek, mis lihtsustab oluliselt asünkroonset programmeerimist. Kuna tegemist on Päästeameti jaoks töövahendiga, kus suurem osa on

[1] Repositooriumi salvestamine oma lemmikute hulka. Kasutatakse kas järjehoidjana või vahel ka repositooriumi meeldimise indikeerimiseks.

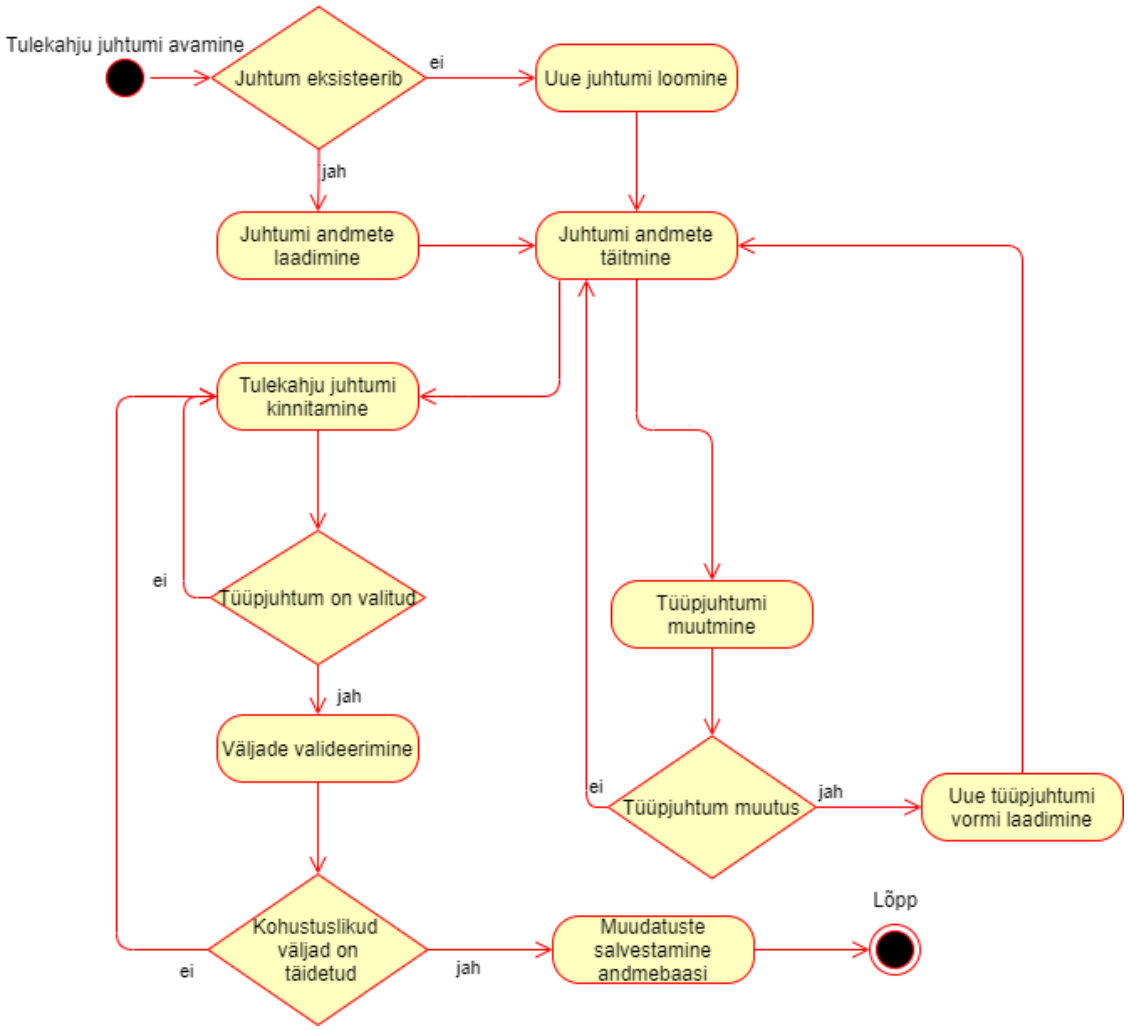
vormide täitmine ja selle info teenuse poolt pärimine või salvestamine, siis lihtsustatud vormide arendamine on väga tähtsal kohal.

Angularis on kasutusel TypeScript, mis teeb koodi selgemaks, arusaadavamaks, veakindlamaks ning vigade esinemisel lihtsamini tuvastatavaks. Erinevalt teisest kahest, on Angularis ka kindlaks määratud piirangud ja juhised rakenduse struktuurile ning seetõttu ei varieeru erinevate arendajate projektid selle poolest nii palju. Raamistikuga juba tuttaval arendajal on seetõttu projektist lihtsam aru saada. Kuna OIS meeskonnast käib läbi palju välisarendajaid, siis selline selgus aitab neil kiiremini projekti sisse elada ja vastupidi ka teistel meeskonna arendajatel hiljem nende loodust paremini aru saada. Osa mooduleid OIS-st on juba ümber kirjutamise järgus või ümber kirjutatud. Tagamaks, et rakendus või vähemalt selle uued moodulid oleks jätkuvalt SPA-d, on selle jaoks loodud Angularis ka uus klientrakendus. Sinna on juba kokku koondatud mõned väiksemad rakenduse moodulid ning ära on tehtud suur eeltöö lihtsustamaks uute moodulite lisamist.

Võttes arvesse töö autori varasemat kogemust ning eelnevat analüüsi, sai otsustatud, et parim variant on kasutusele võtta Angular ja ehitada klientrakendus varem selle jaoks loodud klientrakenduse juurde. Kasutajaliidese ülejäänud rakendusega väljanägemiselt sarnaseks tegemise jaoks on kasutusele võetud vabavaraline mobiilisõbralik CSS raamistik Bootstrap [20].

Tulenevalt vaja minevate aadressiandmete spetsiifikast sai kaardilt aadresside määramiseks valitud Maa-ameti aadressiandmete süsteemi andmeid sisaldav integreeritav aadressiotsingu kasutajaliides In-ADS [21]. Antud liides võimaldab tulekahjujuhtumite aadressobjektide määramist ülejäänud süsteemis kasutatava *ads_oid* täpsusega.

Joonis 3 on kujutatud menetleja põhitöövoo lihtsustatud diagramm.



Joonis 3. Menetleja põhitöövoo diagramm

4.2.3 Tagarakendus

Ohutuse infosüsteemi haldab ettevõtte pool üks meeskond. Kõik selle süsteemiga seotud serveri poolsed rakendused ja mikroteenused on kirjutatud Java programmeerimiskeeles. Seetõttu sai otsustatud ka käesolevas töös loodava lahenduse juures kasutada Javat, et meeskonna arendajad ei peaks ühe mooduli tõttu õppima ära uut programmeerimiskeelt.

Välisvaatlejate tulekahjujuhtumite mitmekordseks avamiseks põhjendust kordamata tuleks juhtumite avamised vahemälus talletada. Võimalus selleks oleks kasutada andmebaasi, aga selline lahendus tekitaks üleliigseid andmebaasi kirjeid. OIS põhirakenduses on juba kasutusel Redis – populaarseim võti-väärtus baasil andmeid mälus hoidev andmehoidla, mida kasutatakse andmebaasina, vahemäluna või sõnumivahetussüsteemina [22]. Probleemi lahendamiseks otsustatigi kasutada Redis-t,

sest antud andmed pole sellised, mida peaks tingimata säilitama rakenduse maha kukkumisel. Lisaks saab ära kasutada juba olemasolevat põhirakenduse andmehoidlat.

Loodava projekti jaoks raamistike valimisel jäid valikusse kaks OIS-s juba varasemalt kasutuses olevat raamistikku: Spring Boot ja Micronaut. Mõlemad on lahenduse jaoks hästi sobivad, kuna lihtsustavad oluliselt REST veebiteenuste loomist.

Spring Boot on lahendus Spring raamistiku [23] keerulisele konfigureerimisprotsessile *convention over configuration*[1] põhimõttel [24]. Arendusprotsess on lihtsustatud, kuna vaja pole eraldi veebiserverit, vaid see on juba raamistikku sisse ehitatud ja automaatselt konfigureeritakse toetatud väliseid teeke [25].

Micronaut on Sping Boot-st uuem JVM raamistik mikroteenuste loomiseks pidades silmas pilvekeskondi. Micronaut on loodud olema väiksema mälukasutuse ja lühema käivitusajaga. See on saavutatud kasutades eraldi konteinerit sõltuvuste sisestamiseks kompileerimise ajal [26].

Mõlemat raamistikku kasutades on loodud sama funktsionaalsusega lihtne testrakendus[2], et võrrelda rakenduste käivitusaega, kokku ehitamisel loodava faili suurust, mälu kasutust ja jõudlust [27]. Võrdlemisel on kasutatud Spring Boot versiooni 2.1.3 [28] ja Micronaut versiooni 1.0.0.RC1 [29] ning testimisel on kasutatud IntelliJ IDEA [30]. Jõudlustesti jaoks saadetakse 40000 POST ja GET päringut testandmetega samaaegselt kahekümne lõime pealt. Tabel 2 on näha, et käivitusaja ja JAR faili suuruste erinevus testrakenduse puhul on märkimisväärne, andes eelise Micronaut-le. Micronaut on ka mälu kasutuse poolest väikese testrakenduse tulemuste järgi efektiivsem. Päringute töötlemise juures pole erinevused niivõrd märkimisväärsed ja pole seotud otseselt raamistikuga, vaid asjaoluga, et kasutab Micronaut vaikimisi baasserverina Netty-t, aga Spring Boot raamistikus on kasutusel Tomcat [27].

[1] Spring raamistiku enamkasutatavad konfiguratsioonid on Spring Boot-is eelkonfigureeritud, et kasutaja saaks rakenduse konfigureerimise asemel tegeleda arendusega.

[2] Testrakendus ja testide läbi viimine pole tehtud diplomitöö autori poolt.

Tabel 2. Micronaut ja Spring Boot testrakenduse tulemused.

	Spring Boot	**Micronaut**
Käivitusaeg	6-7 sekundit	3-4 sekundit
Kokku ehitatud JAR faili suurus	24.2 MB	12.1 MB
Heap mälu kasutus peale käivitust	305 MB	254 MB
Muu mälu kasutus peale käivitust	81 MB	51 MB
Keskmine sekundis töödeldud POST päringute arv	1176	1290
Keskmine sekundis töödeldud GET päringute arv	1428	1538

Mõlemas raamistikus on suur hulk lisateeke, mis lahendavad suurema osa kasutajate vajadustest. Projekti esmaseks loomiseks on Micronaut-il olemas CLI, Spring Boot projekti alustamiseks on aga kasutada Spring Initializr [31]. Mõlemad variandid loovad projektile esmase struktuuri ja failid ning lisavad vastavalt tehtud valikutele vajalikud sõltuvused.

Arvestades Micronaut-i eeliseid saigi esialgu see valitud ja projekti alustatud, aga päris kiiresti tulid välja puudused, mis on hiljuti välja tulnud tehnoloogiate puhul tavalised. Kuigi raamistiku dokumentatsioon on väga põhjalik, siis seal on siiski kirjeldatud ideaalolukorrad. Sattudes projekti edenedes probleemide otsa, ei ole uue tehnoloogia puhul veebist eriti abi leida, mille tõttu kulub hulk aega lahenduseni jõudmiseks.

Teiseks suuremaks probleemiks sai Spring Data [32] sarnase andmebaasiga suhtlemise ja andmete töötlemisega seotud abstraktsiooni tehnoloogia puudumine[1], mis eemaldaks suure osa *boilerplate*-koodist ja võimaldaks keskenduda funktsionaalsuse arendamisele.

SOS rakendus saadab tulekahjujuhtumite andmeid OIS-i läbi Apache ActiveMQ – vabavaraline Java baasil sõnumiserver, mis kasutab AMQP protokolli [33]. Micronaut-il

[1] 31.03.2021 seisuga on Micronaut-i lisandunud Micronaut Data [65], mis lisab selle funktsionaalsuse.

aga puudub ActiveMQ toetus[1]. Probleemi lahendamiseks peaks kasutama suurt hulka Spring raamistiku teeke ja need ühildama Micronaut-ga, mis teeb süsteemi keeruliseks ja võib tulevikus tekitada probleeme.

Kuigi Micronaut on vägagi võimekas raamistik mikroteenuste loomiseks ja areneb võimaluste poolest hoogsalt edasi, sai eelnevalt välja toodud probleemide tõttu valitud loodava rakenduse teenuse poole raamistikuks Spring Boot.

4.3 Arhitektuur

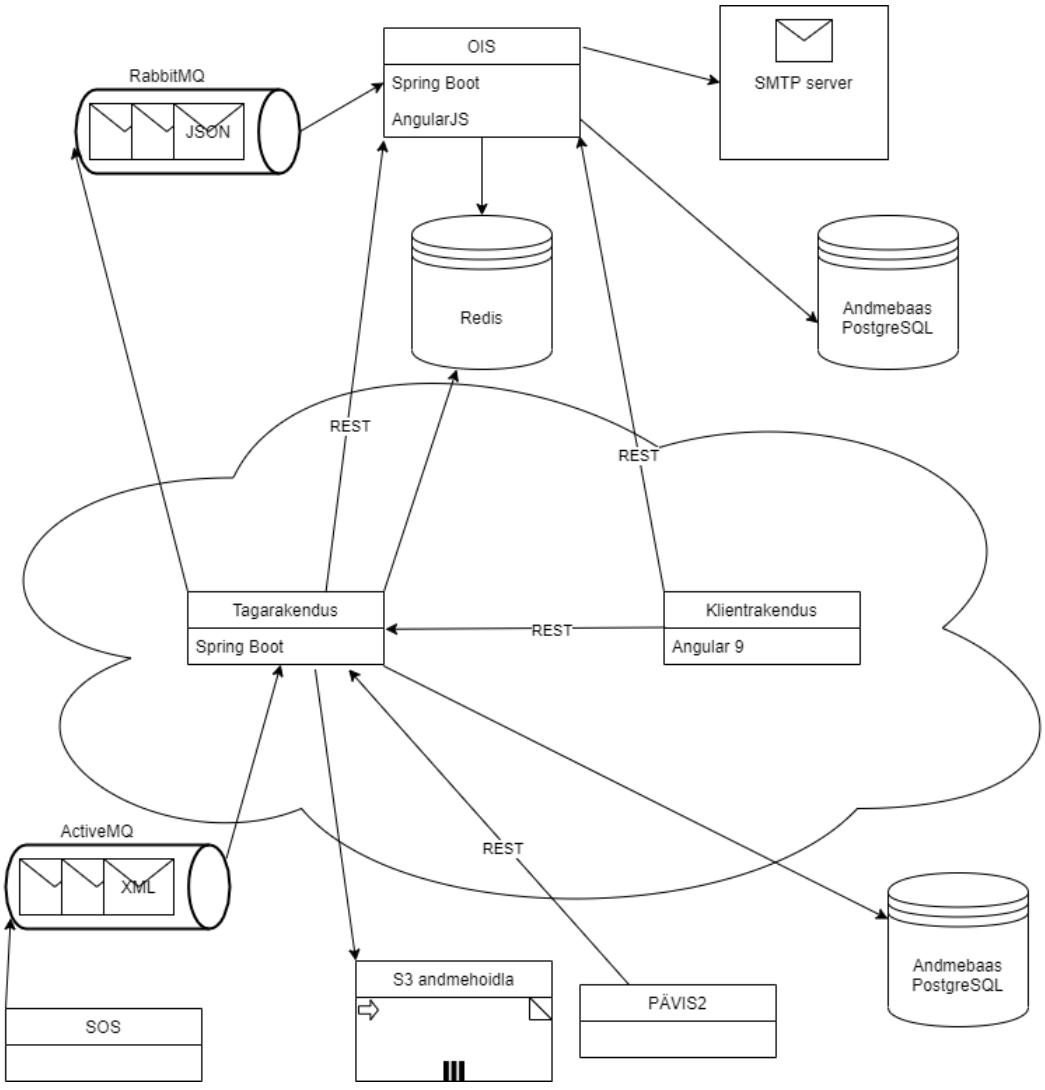
Tulekahjujuhtumite andmed saadetakse loodavasse rakendusse kahest erinevast allikast. SOS saadab andmed XML formaadis ActiveMQ järjekorda (*queue*), kust juhtumite rakendus need vastu võtab ja vastavalt nõuetele töötleb. PÄVIS2 rakendusega suhtlus toimub kasutades REST päringuid. Tulekahjujuhtumi andmed saadetakse POST päringuga JSON formaadis juhtumite rakendusse, kus need nõuetele vastavalt töödeldakse ja rakenduse andmebaasi sisestatakse. Klientrakenduse ja tagarakenduse vaheline suhtlus käib samuti REST päringutega, milles pannakse iga päringuga kaasa autentimisel saadud võti. Loodav rakendus on tihedalt seotud ka põhirakendusega, mille andmebaasis on infot, mida juhtumite rakendus eraldi maha ei salvesta, nagu näiteks kasutajate info, ehitiste info ja tegevuslogi. Nii klientrakenduse kui tagarakenduse suhtlus põhirakendusega käib REST teenustega.

Selleks, et igasse moodulisse ei peaks looma eraldi kogu e-kirjade saatmise funktsionaalsust, on põhirakenduses loodud selle jaoks vastav teenus. Ühendus meiliserveriga luuakse ainult põhirakenduses ja loodav rakendus paneb kokku e-kirja sisu ning saadab vajalikud andmed põhirakendusse. E-kirjade info saatmine toimub läbi RabbitMQ, mis on sarnaselt ActiveMQ-le vabavaraline sõnumivahetusserver [34], aga on autori arvates lihtsam kasutada. Arenduse jaoks sai kasutusele võetud FakeSMTP [35] meiliserver – Javas kirjutatud SMTP serveri rakendus, mis püüab kõik saadetud e-kirjad kinni. Sellel on olemas ka GUI, mis võimaldab hõlpsasti valideerida e-kirja sisu, saatjat,

[1]2021. aastal Micronaut versiooniga 2.3 [67] lisandus Micronaut JMS [66], millega kaasneb ka ActiveMQ toetus.

vastu võtjaid, pealkirja ja saatmise aega. Produktsioonis on kasutusel ettevõtte oma meiliserver.

Loodav klientrakendus ja tagarakendus hakkavad arendus-, test- ja produktsioonikeskkonnas resideeruma ettevõtte Cloud Foundry pilveteenuses [36]. Andmebaas, S3 andmehoidla ja Redis hakkavad olema eraldi serverites. Arenduse jaoks töö autori arvutis jooksevad andmebaasid, S3 andmehoidla, ActiveMQ, RabbitMQ ja Redis Docker-is. Ülevaatlikku pilti loodava rakenduse jaoks vajalike komponentide arhitektuurist ja omavahelisest suhtlusest on näha Joonis 4. Ettevõtte keskkondade üles seadmine ja konfigureerimine jääb selle töö skoobist välja.



Joonis 4. Komponentjoonis

5 Teostus

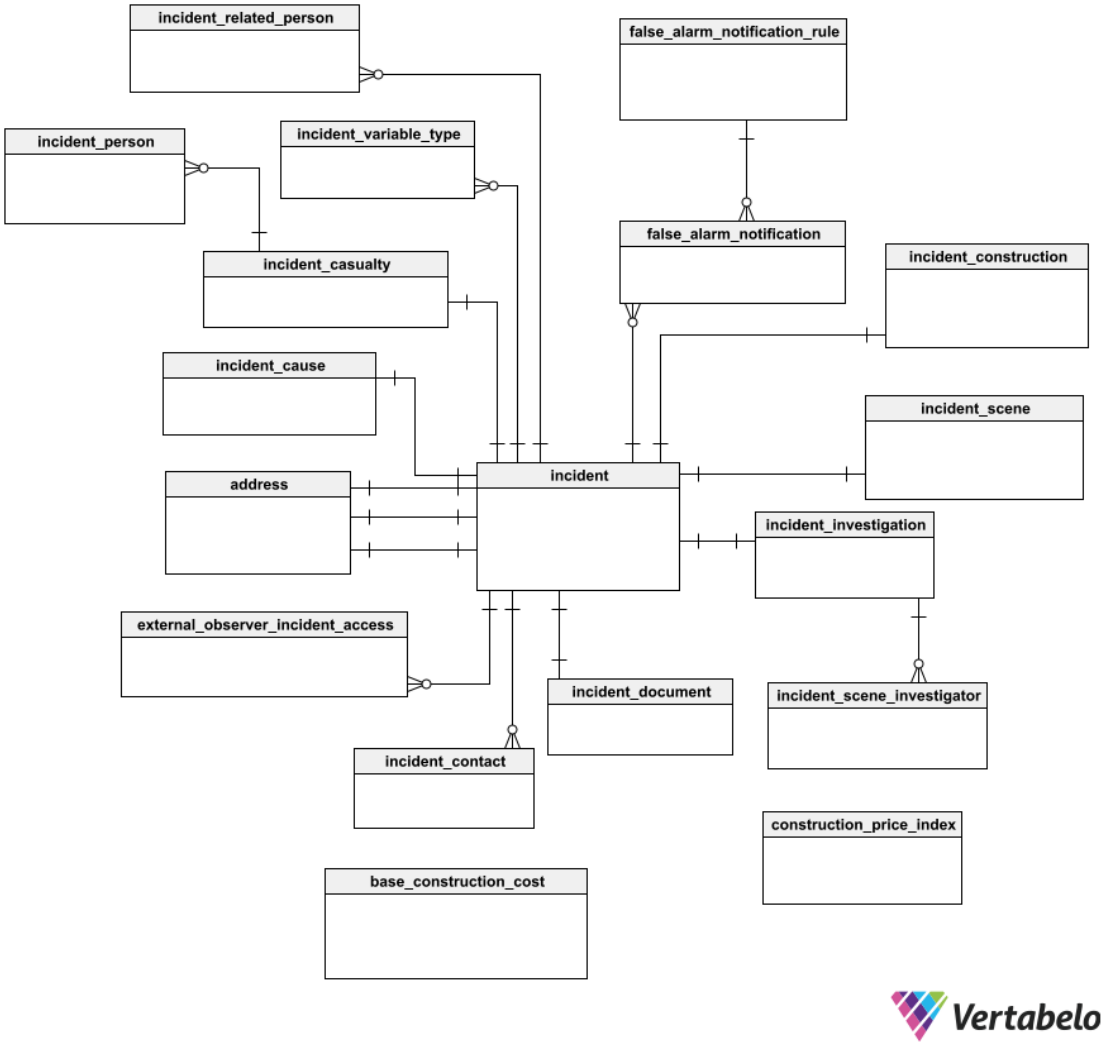
Käesolevas peatükis on kirjeldatud diplomitöö raames loodava veebirakenduse arendusprotsessi ja selle käigus tekkinud probleeme. Kõigepealt vaadeldakse andmebaasi disaini loomist, seejärel tagarakenduse ja klientrakenduse arendusprotsessi ning rakenduse turvalisust. Lõpuks antakse ülevaade rakendusele teostatud testimise poolest.

5.1 Andmebaasi disainimine

Andmebaasi disainimise esimeseks suureks mõttekohaks on vormide küsimustiku ülesehitus, kuna küsimuste hulk on väga suur. Tuleb otsustada, kas teha lihtsustatud variant, kus andmebaasis hoitakse eraldi iga küsimuse vastust, või dünaamiline küsimustik. Esimese variandi puhul tuleks klientrakenduses iga küsimus eraldi välja kirjutada. Teine variant oleks selline, kus küsimused on koos omavaheliste sõltuvuste, küsimuse tüüpide (tekstiväli, kuupäeva väli), järjekorra numbrite ja muude vajalike parameetritega andmebaasis, mis võimaldaks küsimusi baasi iga hetk juurde lisada ilma klientrakendusse suuri muudatusi tegemata. Klientrakenduses oleks üles ehitatud struktuur, mis küsimuse tüübi järgi kuvab kasutajale antud tingimuste järgi küsimused. Lihtsama küsimustiku puhul oleks teine variant väga hea moodus suur hulk koodi kokku hoida. Antud töö puhul oleks selline variant tulnud aga hoopis keerukam ja poleks isegi klientrakenduses suurt koodi kokkuhoidu võimaldanud, kuna küsimuste omavahelisi sõltuvusi ja lisatingimusi oli niivõrd palju. Seetõttu sai valitud esimene variant.

Üheks probleemiks vana lahenduse juures oli kogu andmestiku koondatus ühte suurde tabelisse. Selle vältimiseks jagas töö autor andmed klientrakenduse tulekahjujuhtumite detailvaate alamvaadete järgi eri tabelitesse. Selline jaotus ei tee ka Päästeameti poolt palutavate väljavõtete tegemist keeruliseks. Eraldi klassifikaatorite tabelit loodavasse andmebaasi ei lisatud, sest selline tabel on põhirakenduses olemas. Kõik valikuvariantidega küsimuste klassifikaatorid said lisatud põhirakenduse klassifikaatorite tabelisse, kust klientrakenduses komponendi avamisel vajalikud valikud ülemkoodi järgi sisse laetakse.

Andmebaasi disainimisel tuleb silmas pidada, et tegemist on tulekahjujuhtumi toimumise hetkeseisu säilitamisega. Lisaks tuleb vältida ebavajalikke päringuid põhirakendusse. Seetõttu on põhirakenduses olevate kasutajate ja ehitiste olemitega loodud küll seos ehk tegemist on justkui vahetabelitega, aga loodavas rakenduses säilitatakse siiski rakenduse tööks vajalikke andmeid uues andmebaasis. Loodud olemite semantikad on kirjeldatud Lisas 2. Lihtsustatud olemi-suhte diagrammi on näha Joonis 5 ja täielikku diagrammi saab näha Lisas 3.



Joonis 5. Olemi-suhte diagramm

5.2 Tagarakenduse arendus

5.2.1 Projekti loomine

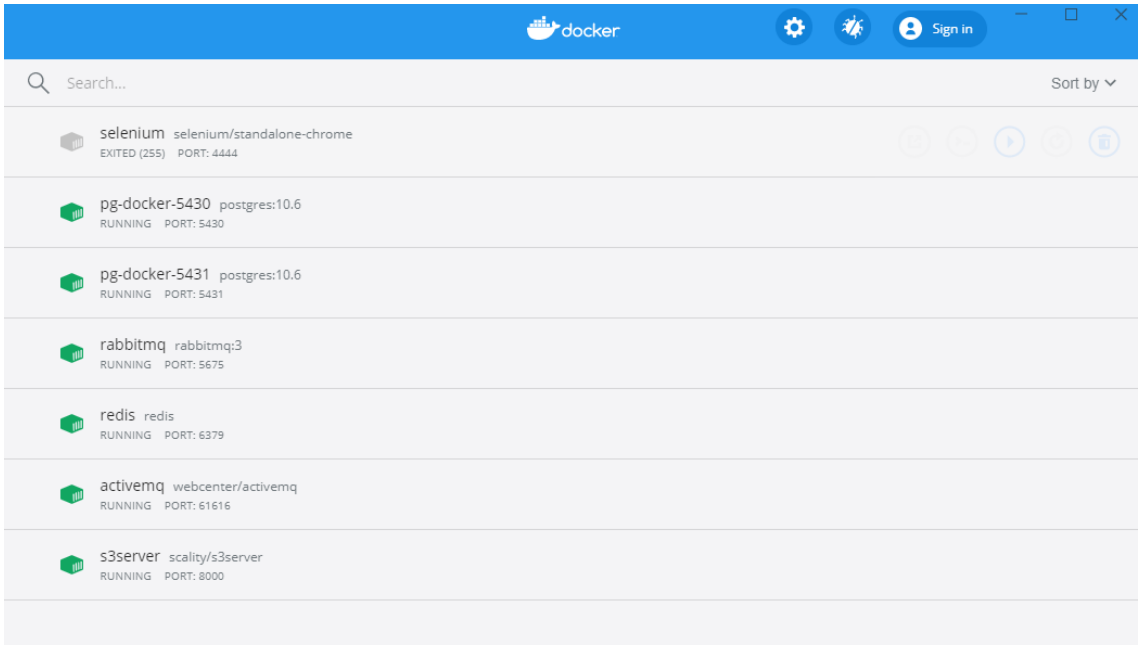
Rakenduse teenuse poole arendusel on projekti esmasel loomisel abiks veebis saada olev tööriist Spring Initializr [31]. Selle kaudu on võimalik valida kasutaja soovidele vastavad tööriistad ja parameetrid nagu projekti arenduskeel ja versioon, Spring Boot-i versioon, ehitustöörist ning esmased sõltuvused. Peale valikute tegemist saab loodud projekti ZIP failina. Loodud failis on olemas projekti esmane struktuur ja vajalikud sõltuvused. Spring Boot-i eeliseks on sisse ehitatud Tomcat server, mis tähendab, et kasutaja ei pea rakenduse serveerimiseks eraldi vaeva nägema. Loodav lahendus kasutab programmeerimiskeelena Java 11 ja raamistikuks on valitud Spring Boot versioon 2.2.4.RELEASE. Ehituse ja sõltuvuste halduse tööriistaks on Gradle, mis aitab automatiseerida tarkvara ehitusprotsessi. Sõltuvuste lisamiseks tuleb soovitud valik lisada *build.gradle* faili ja tööriist hoolitseb selle eest, et kõik lisatud sõltuvused ja omakorda nende sõltuvused oleksid olemas. Väljavõtet loodava projekti vastavast failist näeb Lisas 4. Nagu analüüsi osas on välja toodud, lihtsustab Spring Boot oluliselt projekti konfigureerimist. Raamistik on välja töötatud põhimõttel, et enam kasutatavad konfiguratsioonid on kasutaja jaoks vaikimisi sisestatud. Suurema osa konfiguratsioonide muutmiseks piisab *application.yml* failis vastavate sätete lisamisest, mille väljavõtet antud projektis saab näha Lisas 5.

5.2.2 Välised komponendid

Loodava rakenduse töö sõltub suuresti lahenduse realiseerimiseks vajalikest välistest komponentidest ja teenustest nagu andmebaas, ActiveMQ, S3 ühilduv andmehoidla ja Redis. Loetletud komponentide üles seadmiseks on kasutusel Docker [37], mis võimaldab läbi virtualiseerimise luua isoleeritud konteinereid vajalike rakenduste ja teenuste jooksutamiseks. Lisaks on arenduse jaoks kasutusele võetud Docker Desktop [38], mis lubab kasutajal käsurea liidese asemel konteinereid hallata läbi graafilise liidese, aga võimaldab ka hõlpsasti Docker-i konfiguratsioone muuta. Joonis 6 on näha loodava rakenduse jaoks loodud ja käima pandud Docker-i konteinereid ning nendes jooksvaid *image*-d[1]. Andmebaasi jaoks kasutatud *image* on postgres:10.6, Redis jaoks on kasutusel

[1] Hetkeseis vajalikust rakendusest nagu näiteks andmebaas, mille saab konteineris käivitada ja siis teenusena kasutada

redis:latest [39], ActiveMQ jaoks webcenter/activemq:latest [40] ja S3 ühilduva andmehoidla jaoks scality/s3server [41].



Joonis 6. Välised komponendid Docker-is

5.2.3 Teenuse API

Spring Boot raamistik võimaldab hõlpsasti luua REST API teenuseid kasutades selleks annotatsioone nagu @RestController ja @GetMapping. Neid teenuseid kasutades suhtlevad klientrakendus ja OIS põhirakendus loodava tagarakendusega. Tulekahjujuhtumi kontrolleri pagineeritud nimekirja päringu meetodi koodi võib näha Joonis 7. Kõik rakenduse API ressursid on välja toodud Lisas 6.

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/incident")
public class IncidentController {

    private final IncidentService incidentService;

    @Secured({"ROLE_KASUTAJA"})
    @PostMapping("/page")
    public Page<IncidentListItemDto> page(@RequestBody final IncidentSearch
    search){
        Page<Incident> incidents = incidentService.getIncidents(search);
        return incidents.map(incidentService::getListItemDetails);
    }
```

Joonis 7. Teenuse REST API koodinäide

5.2.4 Andmebaas ja domeeniobjektid

Andmebaasi ühenduse seadistamine käib läbi Lisas 5 kirjeldatud konfiguratsioonifaili. Tagamaks iga rakenduse keskkonna ja uute arendajate andmebaasi struktuuri ühtsust on kasutusele võetud avatud lähtekoodiga migratsioonihalduse tööriist Flyway [42], mis tagab andmebaasi versioonihalduse. Selle kasutamiseks on vaja konfiguratsioonis anda ette migratsioonifailide kaust, kus hakkavad olema kindla nimeformaadiga SQL migratsioonifailid. Vaikimisi skeemi asendamiseks uuega on vaja Spring Boot-is Flyway-d rohkem seadistada. Seadistuse kood on välja toodud Joonis 8.

```
@Bean
public Flyway flyway() throws Exception {

    Flyway flyway = Flyway.configure()
            .dataSource(dataSource())
            .schemas("syndmus") //Automatically creates schema
            .baselineOnMigrate(false)
            .outOfOrder(false)
            .load();
    flyway.repair();
    flyway.migrate();
    return flyway;
}
```

Joonis 8. Flyway skeemi muutmise kood

Andmebaasiga suhtlemiseks on rakenduses kasutusel Spring Data JPA raamistik [43], mis lihtsustab oluliselt JPA baasil repositooriumite loomist vähendades arendaja poolt kirjutatava koodi hulka ning see tuleb koos mitmete lisadega nagu pagineerimine ja auditeerimine. Isegi keerukamate andmepäringute jaoks pole tingimata vaja kirjutada ridagi SQL koodi. Koodinäide tulekahjujuhtumi repositoorimist on Joonis 9.

```
@Repository
public interface IncidentRepository extends JpaRepository<Incident, Long>,
JpaSpecificationExecutor<Incident> {
    Incident findByIncidentNumber(IncidentNumber incidentNumber);

    @Query(value = "SELECT max(i.incident_number) FROM incident i WHERE
    i.is_manually_created = true", nativeQuery = true)
    Long getMaxManuallyCreatedIncidentNr();

    List<Incident>findAllByIncidentConstructionConstructionIdOrderByOccurredA
    tDescCreatedAtDesc(Long constructionId);
}
```

Joonis 9. Tulekahjujuhtumi repositooriumi kood

Lahenduses on kasutusel ka Lombok projekti teek [44], mis võimaldab läbi annotatsioonide genereerida konstruktoreid, gettereid ja settereid ning muud vähendades

arendaja poolt kirjutatava koodi hulka ja muutes seeläbi koodi lihtsamini loetavaks. Iga andmebaasi tabeli jaoks on loodud domeeniobjekt, millele tuleb *@Entity* annotatsioon lisada, et raamistik oskaks selle vastava tabeliga seostada. Dokumendi domeeniobjekti kood on näha Lisas 7. Klassifikaatori objektide andmebaasi salvestamiseks ja pärimiseks on loodud konverter, mille kood on Lisas 8. Kõigi salvestatavate objektide jaoks ei ole mõttekas luua eraldi tabelit andmebaasi. Sellised objektid, mis saadetakse PÄVIS2 poolt, aga loodavas rakenduses muutmisele ei kuulu ja kasutatakse ainult info kuvamiseks, salvestatakse baasi JSONB [45] formaadis. Sellesse formaati andmete salvestamiseks ja pärimisel Java objektiks konverteeriseks on kasutusel vladmihalcea/hibernate-types-52 [46] teek. Koodinäide kasutusest on Joonis 10.

```
@Type(type = "com.vladmihalcea.hibernate.type.json.JsonBinaryType")
private List<Hydrant> extinguishersHydrants = new ArrayList<>();
```

Joonis 10. Hibernate-types-52 teegi kasutuse koodinäide JSONB formaadi jaoks

5.2.5 SOS juhtumite vastu võtmine

Tulekahjujuhtumid saadetakse SOS-st ActiveMQ sõnumiserveri järjekorda. Sealt andmete vastu võtmiseks kasutatakse Spring JMS integratsiooni raamistikku [47], mis lihtsustab JMS API kasutust. Sõnumiserveri jaoks on rakenduse konfiguratsioonifailis seadistatud ActiveMQ ühenduse ja sõnumijärjekorra parameetrid. Saadetavad andmed on XML formaadis ning nende andmete töötlemiseks ja Java objektideks konverteerimiseks on kasutusele võetud JAXB teek [48]. Lühendatud kood sõnumi vastu võtmiseks on Lisas 9.

5.2.6 PÄVIS2 juhtumite vastu võtmine

PÄVIS2 sõnumid saadetakse JSON formaadis ja nende vastu võtmiseks on REST kontroller, mille kood on Joonis 11. Rakenduste vaheliseks autentimiseks on kasutusel Spring Security Basic Authentication, kus autentimine käib eelnevalt kokku lepitud võtme alusel [49]. Juhtumi salvestamisele või muutmisele eelneb suur hulk tingimuste kontrolle, mille järel tehakse kindlaks tüüpjuhtum. Sõnumiga saadetud juhtumi andmed tuleb üle kontrollida ja saadud klassifikaatorite koodid ümber muuta ohutuse infosüsteemis kasutatavateks koodideks, millest suurem osa kattuvad. Lisas 10 on näha juhtumi vastu võtmise põhimeetodi kood.

```
@RestController
@RequestMapping("/pavis")
@AllArgsConstructor
public class PavisController {

    final PavisCallReceiver pavisCallReceiver;

    @PostMapping("/event")
    public void receiveEvent(@RequestBody Event event) {
        pavisCallReceiver.createOrUpdateIncidentFromEvent(event);
    }
}
```

Joonis 11. PÄVIS2 sõnumi vastu võtmise kontroller

5.2.7 Dokumentide lisamine

Kasutaja poolt lisatud dokumentide kohta luuakse andmebaasi kirje dokumendi infoga ning seejärel salvestatakse dokumendi sisu S3 tüüpi andmehoidlasse, kus infona on kaasas andmebaasi kirje id, mille järgi dokumente hiljem alla saab laadida. Andmehoidlaga suhtlemiseks on kasutatud Amazoni aws-java-sdk-s3 teeki [50], milles on olemas vajalikud meetodid rakenduse ja andmehoidla vaheliseks autentimiseks ja ühenduse loomiseks, dokumentide salvestamiseks ning alla laadimiseks. Koodi faili laadimiseks andmehoidlasse näeb Joonis 12. Ülejäänud S3 teenuse kood on näha Lisas 11.

```
public void uploadFile(Long id, String pathName, InputStream inputStream,
Long fileSize) {
    ObjectMetadata objectMetadata = new ObjectMetadata();
    objectMetadata.setContentLength(fileSize);
    s3client.putObject(
            config.getBucketName(),
            pathName + "/" + id,
            inputStream, objectMetadata
    );
}
```

Joonis 12. S3 andmehoidlasse faili üles laadimise kood

5.2.8 PDF genereerimine

Tulekahjujuhtumist PDF-i genereerimiseks on põhirakenduse poole arendatud vastav teenus, mis kasutab wkhtmltopdf tööriista HTML dokumendi PDF formaati konverteerimiseks [51]. Nagu PÄVIS2 rakendusega, on põhirakendusega autentimiseks kasutatud Basic Authentication. Dokumendi saatmisel klientrakendusest teenuse poole luuakse HTML *template* ehk malli järgi juhtumi andmete põhjal dokumendi HTML fail. Selleks on kasutusel Thymeleaf SpringTemplateEngine [52]. Fail saadetakse edasi

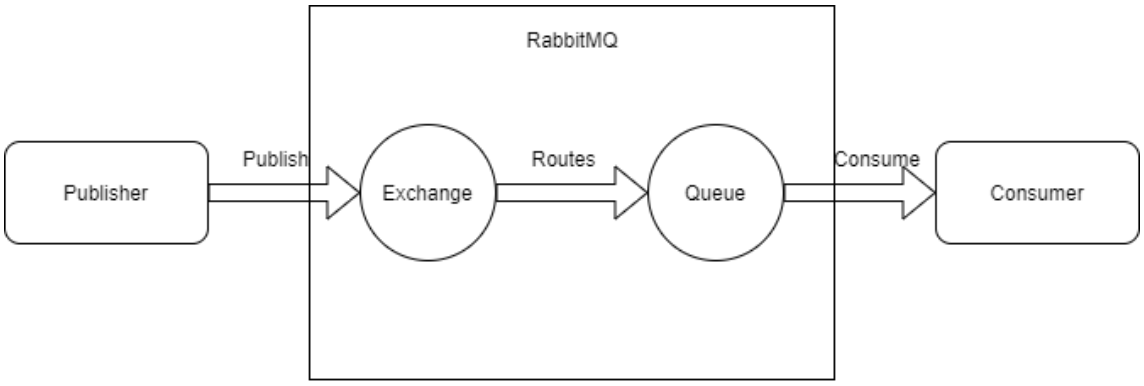
põhirakendusse, kus luuakse selle põhjal PDF ning saadetakse baitide jadana tagasi loodavasse rakendusse ning sealt omakorda klientrakendusse.

5.2.9 CSV väljavõtte loomine

Diplomitöö skoobi raames on CSV väljavõte lahendatud lihtsustatud kujul, mis tähendab, et kaasatud pole kõiki juhtumi andmevälju. Väljavõtte tegemiseks on kasutatud Ostermillerutils ExcelCSVPrinter Java teeki [53], mille kasutamiseks on vaja ära määrata loodava faili päised ning seejärel lähteandmetest koostada faili read viies andmeväljad ja päised vastavusse. Lahenduse kood on Lisas 12. Juhtumi andmetes on valikvastustega küsimuste vastused klassifikaatori koodidena, aga CSV failis on vaja näidata kasutajale koodi asemel tähendusega teksti. Selleks on vaja põhirakendusest laadida loodavasse rakendusse klassifikaatorid *labeli* ehk tähendusega. Vältimaks iga kord klassifikaatorite uuesti laadimist säilitatakse klassifikaatorid rakenduse mälus taaskasutuseks.

5.2.10 E-kirja saatmine

Igas OIS moodulis ei arendata eraldi välja kogu e-kirjade saatmise funktsionaalsust. Ühendus meiliserveriga luuakse vaid OIS põhirakenduses ning sinna edastatakse saatmisele kuuluvate e-kirjade andmed läbi RabbitMQ sõnumivahetustarkvara. Sõnumiserveriga ühenduse seadistamine käib läbi rakenduse konfiguratsioonifaili ning konkreetsem konfiguratsiooni kood on Lisas 13. Sõnumite saatmise voogu läbi RabbitMQ selgitab Joonis 13 [54].



Joonis 13. Sõnumi voog läbi RabbitMQ

5.2.11 Varakahju ja ära hoitud varakahju arvutus

Varakahju ja ära hoitud varakahju arvutus on Päästeameti statistika tähtis osa. Arvutuste valemid ja metoodikad on välja töötatud Sisekaitseakadeemia poolt. Valemid koosnevad

mitmest kasutaja poolt täidetavast sisendparameetrist, millest üheks on ruutmeetri baasmaksumus. Ehitised on kasutusotstarvete järgi jagatud gruppidesse ning igale kasutusotstarbele on määratud kindel ruutmeetri baasmaksumus, mis on sisestatud tabelisse base_construction_cost. Teine suurem komponent on ehitushinna indeks, mille arvutamiseks on tarvis Statistikaameti andmebaasist pärida ehitushinna koondindeksi muutus võrreldes eelmise aastaga. Päringu vastus salvestatakse loodava rakenduse andmebaasi tabelisse construction_price_index, mida vajadusel uuendatakse. Vastavalt kasutaja sisestatud väärtustele arvutatakse varakahju ja ära hoitud varakahju väärtused eurodes. Arvutuseks kasutatud meetodite ja valemite täpsema sisuga on võimalik tutvuda Sisekaitseakadeemia Digiriiulis [55].

5.2.12 Välisvaatleja juhtumi avamise lahendus

Tulekahjujuhtumite raames on välisvaatleja roll mõeldud partnerasutuse töötajatele oma seadusest tulenevate ülesannete täitmiseks. Antud lahendus on loodud auditeeritavuse tagamiseks. Kasutaja mugavusele mõeldes tuleb tagada, et 24 tunni jooksul sama juhtumit avades uuesti põhjust sisestama ei peaks. 24 tunni jooksul tehtud esmakordsed juhtumite avamised salvestatakse statistika jaoks tabelisse external_observer_incident_access ning iga kasutaja konkreetse tulekahjujuhtumi avamised salvestatakse ka Redis-e vahemällu. Selleks on kasutatud Spring Data Redis raamistikku [56], mis hoiab andmeid mälus võti-väärtus formaadis. Loodavas rakenduses koosneb võti eesliitest, juhtumi id-st ja kasutaja isikukoodist ning kirje aegumise ajaks on määratud 24 tundi. See tagab iga kasutaja ja juhtumi kohta unikaalse võtme, mille järgi on võimalik kindlaks teha, kas kasutaja avab juhtumit 24 tunni jooksul esimest korda. Kood vastavate kirjete kätte saamiseks ja loomiseks on välja toodud Lisas 14.

5.3 Klientrakenduse arendus

Tulekahjujuhtumite moodul lisandub varasemalt olemas oleva klientrakenduse juurde, mis on loodud Angular 9 raamistikku kasutades. Angular võimaldab rakenduse osad iseseisvateks mooduliteks jagada ning määrata URL-i segmendid, millele navigeerides vastav moodul laetakse, säilitades samal ajal üheleherakenduse omadused. Koodinäide selle saavutamiseks on Joonis 14.

```
{
    path: 'incidents',
    component: IncidentModuleComponent,
    loadChildren: './incident/incident.module#IncidentModule',
    runGuardsAndResolvers: 'always',
    canLoad: [IncidentModuleGuard]
},
```

Joonis 14. Tulekahjujuhtumi mooduli URL-i segmendi määramine

Rakenduse stiliseerimiseks on kasutusel Bootstrap, mida on vastavalt põhirakenduse stiilile kohandatud. Kasutamiseks on vaja NPM kasutades rakendusele installeerida Bootstrap ning seejärel importida see angular.json failis. Sarnaselt tagarakendusele, kus väliste teekide haldamiseks on kasutusel Gradle, on klientrakenduses selle jaoks kasutusel NPM.

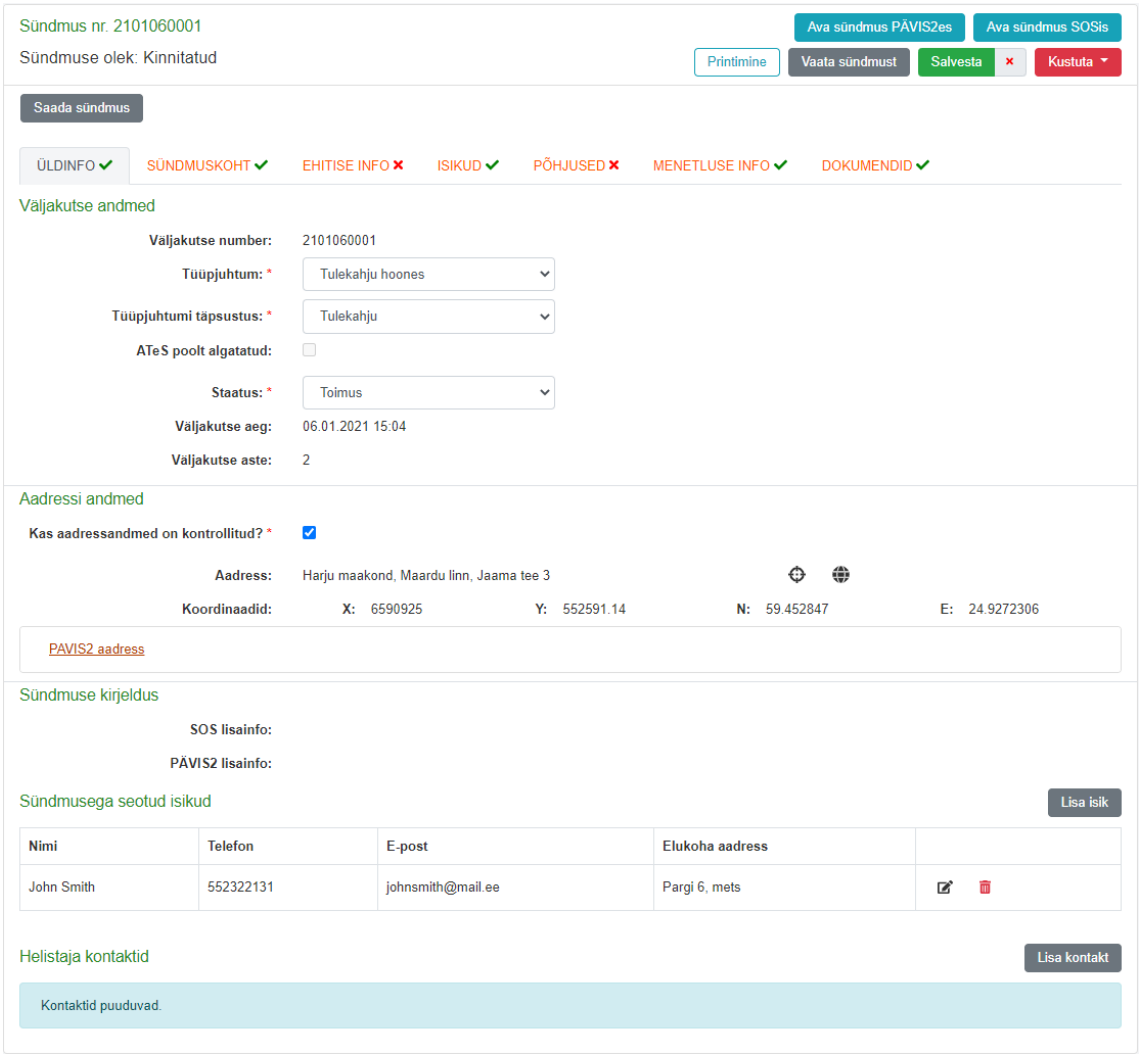
5.3.1 Komponendid ja vaated

Angular raamistikus on vaated ja selle osad jagatud komponentideks, mis on taaskasutatavad vähendades korduvat koodi. Lihtsustatult koosneb komponent vaate mallist ehk HTML failist ja loogikat sisaldavast TypeScript failist, kuigi mõlemad võivad olla ka ühes failis. Kasutates Angular-i süntaksit võimaldab raamistik kahesuunalist andmete sidumist, kus loogika osas tehtud muudatused objektidele kajastuvad kasutajale kuvatavas vaates kasutades ära DOM manipulatsiooni. Angular CLI on lihtne moodus uute komponentide genereerimiseks. Loomisel luuakse komponendi vajalikud failid ja komponendi tööks vajalik esialgne kood. Käsk komponendi loomiseks läbi Angular CLI on Joonis 15.

```
ng generate component incident-list –inlineStyle --skipTests
```

Joonis 15. Komponendi loomine läbi Angular CLI

Loodavas rakenduses on kasutaja jaoks kaks põhivaadet. Üheks põhivaateks on tulekahjujuhtumite otsingu tulemuste nimekiri ning teine on juhtumi detailvaade, mis omakorda on jagatud seitsmeks alamvaateks, milleks on üldinfo, sündmuskoht, ehitise info, isikud, põhjused, menetluse info ja dokumendid. Joonis 16 on näha rakenduse detailvaate üldinfo alamvaadet. Juhtumite otsingu nimekirja vaade on välja toodud Lisas 15.



Joonis 16. Üldinfo vaade

Vaadete vahel navigeerimiseks on Angular-is kasutusel *routing* ehk marsruutimine. Üheleherakendusena ei laeta iga kasutaja vaate jaoks serverist uut lehte, vaid vaadete vahel liikumisel tuuakse nähtavale või peidetakse vaate jaoks vajalikke komponente. Angular Router [57] tõlgendab brauseri URL-i instruktsioonina vaate vahetamiseks ja vajalike komponentide näitamiseks. Projekti marsruutimise detailid on välja toodud Lisas 16. Erinevate alamvaadete komponentide laadimised on tehtud URL-i *subTab* segmendi järgi. Lühendatud näide sellekohasest koodist on toodud Joonis 17.

```
<app-incident-casualties class="tab-pane active" id="CASUALTIES"
       *ngIf="subTab  === 'CASUALTIES'"
       (componentLoaded)='casualtyLoaded($event)'
       [incident]="incident"
       [incidentForm]="incidentForm"
       [submitted]="submitted"
       [readOnly]="readOnly">
</app-incident-casualties>
```

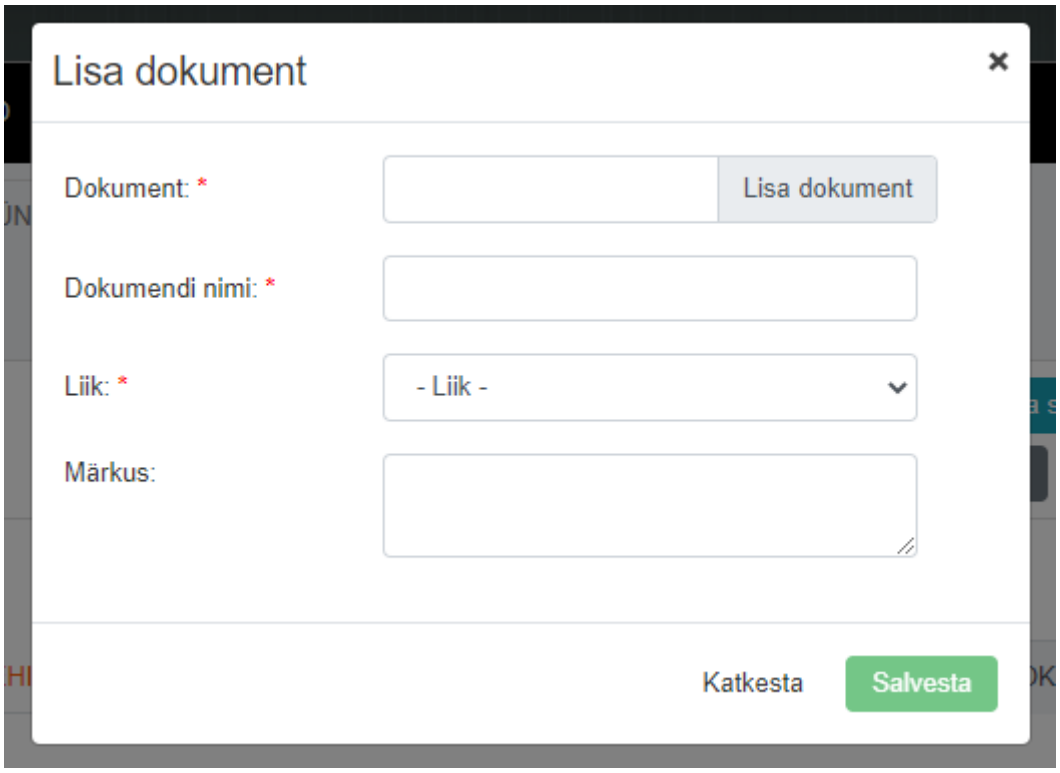
Joonis 17. Isikute komponendi laadimise tingimus

Igal kasutaja rollil pole õigust tulekahjujuhtumeid näha või muuta. Komponentide ja moodulite laadimisel on võimalik lisada tingimuste kontrollid. Selle jaoks on Angular Route Guards [58], mis lisatakse kindlate marsruutide määramisel. Loodavas rakenduses on sellised kontrollid lisatud kogu tulekahjujuhtumite mooduli laadimisele kui ka juhtumi detailvaate vaatamise ja muutmise marsruutidele. Näide juhtumi vaatamisele lisatud kontrollist on Lisas 17.

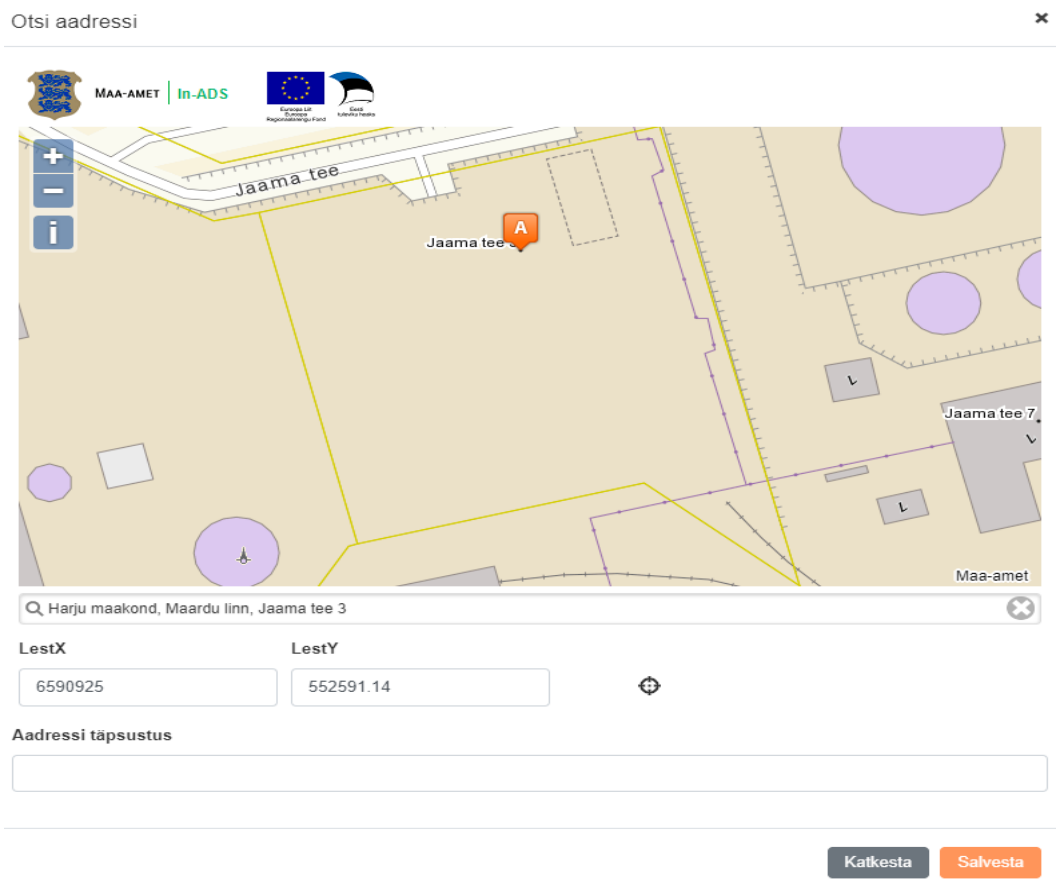
Väiksemate valikuliste vormide, mis pole põhivormide osad, täitmiseks on kasutusel modaalid. Modaal on tavaline Angular komponent, aga avaneb lisaaknana põhivaate peale ja ei ole seotud brauseri URL-ga. Lisaks Bootstrap-le on rakenduses kasutusel ka NgBootstrap, mille API on loodud Angular-i silmas pidades, kust on modaalide jaoks kasutusel NgbModal [59]. Joonis 18 on välja toodud dokumendi lisamise modaali kood ning Joonis 19 on pilt dokumendi lisamise modaalist rakenduses. In-ADS kaardiplugina kasutuse vaade on nähtaval Joonis 20.

```
addDataFile(): void {
    const modalInstance = this.modalService.open(DocumentAddModalComponent, {
      backdrop: 'static'});
    modalInstance.componentInstance.incident = this.incident;
    modalInstance.result.then((document: IncidentDocument) => {
      if (this.incident.incidentDocuments == null) {
        this.incident.incidentDocuments = [];
      }
      this.incident.incidentDocuments.push(document);
      this.toastr.info(TOASTR_MESSAGES.INFO.DOCUMENT_UPLOAD_IN_PROGRESS);
      this.doPolling(document.id);
    }, (error) => {
      console.error(error);
      this.toastr.error(TOASTR_MESSAGES.ERROR.DOCUMENT_UPLOAD_FAILED);
    });
}
```

Joonis 18. Modaali kasutamise kood



Joonis 19. Dokumendi lisamise modaal



Joonis 20. In-ADS kaardiplugina kasutuse modaal

5.3.2 Vormide loomine

Vormidega ümber käimiseks pakub Angular kahte erinevat lähenemist [60]. *Reactive forms*, milles kasutusel olevale vormi objektile on otsene ligipääs. Selliselt ehitatud vormid on rohkem skaleeruvad, taaskasutatavad ja testitavad. *Template-driven forms* toetub mallis kasutatavatele direktiividele otseselt manipuleerimaks vaadeldavat objekti ennast. Autor on käesolevas projektis kasutanud esimest varianti, kuna vormid on klientrakenduse põhiosa.

Vormide funktsionaalsuse kasutamiseks on vaja vastavasse projekti moodulisse importida FormsModule. Vormide ehitamiseks on Angular-is erinevaid mooduseid, aga käesolevas projektis on kasutatud FormBuilder komponenti, millega luuakse vormi grupid ja väljad esialgsete väärtuste ja validaatoritega. Joonis 21 on esitatud kood helistaja kontaktide vormi loomiseks FormBuilder-t kasutades ning vastava vormi HTML kood on nähtav Lisas 18.

```
private buildForm(): void {
  this.callerForm = this.fb.group({
    name: [this.caller.name ? this.caller.name : null, Validators.required],
    phoneNumber1: [this.caller.phoneNumber1 ?
                   this.caller.phoneNumber1 : null,
                  [Validators.pattern('^\\+?[0-9 ]*'),
                   Validators.maxLength(15), Validators.required]]
  });
}
```

Joonis 21. FormBuilder-i kasutamine vormi loomiseks

5.3.3 Andmete laadimine ja salvestamine

Vaate andmete laadimist on võimalik teostada mitmel erineval moel. Andmete laadimiseks vastava komponendi laadimise hetkel on kasutatud ngOnInit meetodit, mis on üks Angular-i pakutavatest elutsükli meetoditest ja välja kutsutakse see vahetult peale komponendi ja andmetega seotud elementide laadimist [61]. Teise moodusena on kasutatusel *Resolver*. Selle kasutamine määratakse ära marsruutimise komponendis, mida on näha Lisas 16. Raamistik hoolitseb vastava komponendi laadimisel selle välja kutsumise eest. Koodinäide *Resolver*-i kohta on Joonis 22.

```
@Injectable()
export class IncidentResolver {
  constructor(private incidentService: IncidentService,
              private router: Router) {}

  resolve(
    route: ActivatedRouteSnapshot
  ): Observable<Incident> {
    if (this.router.getCurrentNavigation().extras.state?.loaded) {
      return;
    }

    return this.incidentService.getIncident(+route.params.incidentId);
  }
}
```

Joonis 22. Tulekahjujuhtumi Resolver-i kood

Tagarakenduse REST API-ga suhtlemiseks kasutatakse teenuse klasse, mille sisestamise eest hoolitseb raamistik kasutades *dependency injection*-t. Tehtavad päringud on asünkroonsed kasutades selleks RxJS teeki ja *Observable*-id, mis kujutavad endast välja kutsutavaid tulevaste väärtuste või sündmuste kogumit [62]. Tulekahjujuhtumite teenuse klassi ja meetodite lühendatud kood on välja toodud Lisas 19. *Observable* kasutamise kood helistaja kontakti kustutamise näitel on näha Joonis 23.

```
this.incidentService.deleteCaller(this.incident, caller)
  .subscribe(() => {
    this.incident.incidentMain.callers.splice(index, 1);
    this.toastr.success(
      TOASTR_MESSAGES.SUCCESS.DELETE_CALLER_CONTACT_SUCCESS
    );
}, (error) => {
  this.toastr.error(TOASTR_MESSAGES.ERROR.GENERAL_SOMETHING_WENT_WRONG);
  console.error(error);
});
```

Joonis 23. Observable kasutamise kood helistaja kontakti kustutamise näitel

5.4 Turvalisus

Turvalisus on kahtlemata iga infosüsteemi väga tähtis osa. Üks suur eelis nii klientrakenduses kui ka tagarakenduses suuremate raamistike kasutamisel on juba raamistikku sisse ehitatud turvalisus ja selle võrdlemisi lihtne kasutamine. Sellega kaasneb aga alati ka risk, kuna avastatud ja avalikuks tulnud turvanõrkusi saab ära kasutada, kui on teada mis raamistiku versiooni rakendus kasutab. Seetõttu tuleks alati veenduda, et kasutatavad raamistikud ja teegid on uuendatud.

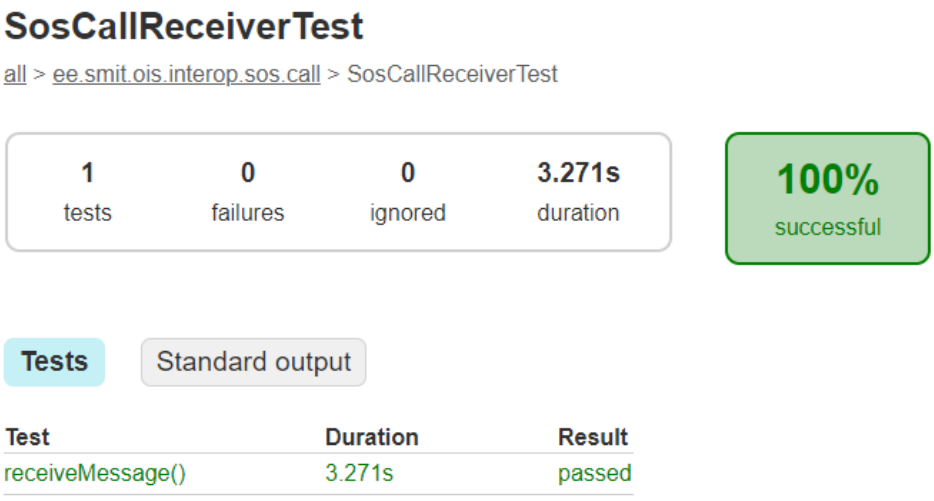
Loodud rakenduses käib kasutajate autentimine läbi UAA. Kuigi autentimine ja autoriseerimine on väga suur osa rakenduse tööst ja töötav lahendus on olemas, siis detailsem kirjeldus jääb käesoleva diplomitöö skoobist välja. Üheks põhjuseks on ettevõtte soov sellise informatsiooni avalikustamist vältida. Teiseks, kõik autentimise implementeerimisel kasutatud koodist pole töö autori kirjutatud, vaid osaliselt on kasutatud varasemalt OIS-i lisatud moodulite autentimise lahendusi.

5.5 Testimine

SOS sõnumi vastu võtmise kohta on loodud automaattest ning PÄVIS2 sõnumite vastu võtmise jaoks on testkeskkonna vastu tehtud koormustestid, mida on selles peatükis kirjeldatud.

5.5.1 SOS sõnumi automaattest

Loodavas rakenduses on sõnumite vastu võtmiseks SOS rakendusest loodud automaattest kasutades JUnit [63] raamistikku. Päringu sisu, mis on näha Lisas 20, võetakse XML failist ning saadetakse seejärel ActiveMQ-sse, kust see rakenduse poolt vastu võetakse, töödeldakse ja andmebaasi salvestatakse. Testi käigus kontrollitakse, kas vastavad andmed on korrektselt andmebaasis olemas. Vastava testi kood on näha Lisas 21 ning käivituse positiivne tulemus on näha Joonis 24.



Joonis 24. SOS sõnumi vastu võtmise testi tulemus

5.5.2 PÄVIS2 sõnumite vastu võtmise koormustest

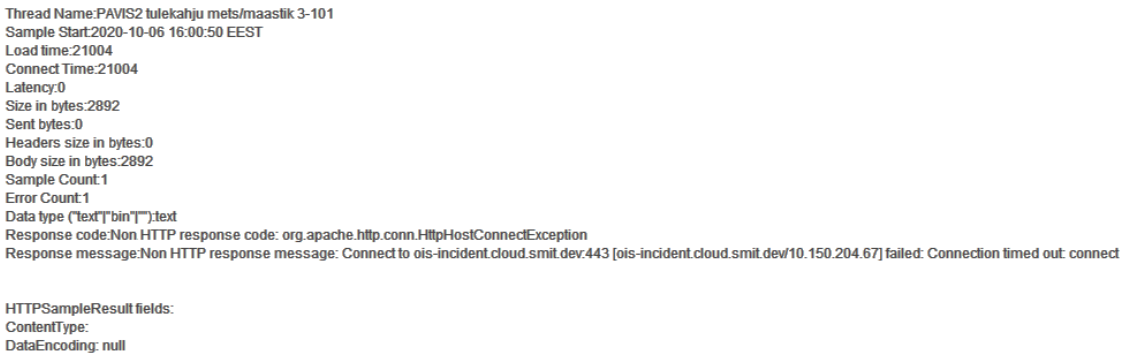
Loodava rakenduse jõudluse testimiseks PÄVIS2 rakendusest sõnumite vastu võtmiseks on läbi viidud koormustestid kasutades Apache JMeter 5.2.1 rakendust [64]. Testid on läbi viidud vastu ettevõtte testkeskkonda. Koostatud on üheksa erinevat testpäringut esindamaks igat tulekahjujuhtumi tüüpjuhtumit, mille juhtumite numbrid genereeritakse suvaliselt. Kõigi testide käivitusel on korduste arvuks määratud kaks kordust. JMeter rakenduses esindavad *thread group*-d kasutajaid.

Esimese testi käivitamisel on kasutajate arvuks määratud iga päringu kohta 2 kasutajat, mis tähendab, et korraga proovitakse saata 18 päringut ning korduste arvu arvesse võttes tehakse kokku 36 päringut. Teises testis on kasutajate arv suurendatud 20 peale, mis tähendab kokku 360 päringut. Kolmandal käivitusel tõstetakse kasutajate arv 200 peale, mis teeb kokku 3600 päringut. Tabel 3 on kajastatud testide käivituste tulemused.

Tabel 3. PÄVIS2 andmete vastu võtmise koormustesti tulemused

Päringute arv	**Minimaalne/Keskmine/Maksimaalne päringu kestvus (ms)**	**Ebaõnnestunud päringuid (%)**	**Läbilaskevõime (päringut/s)**
36	294/848/2359	0.00	9.27
360	441/10946/20990	0.00	13.5
3600	64/59623/112858	27,14	20.9

Tabel 3 tulemustest on näha, et esimese kahe käivituse juures ei tekkinud ühtegi ebaõnnestunud päringut, aga 3600 päringu juures ebaõnnestus umbes 27% päringutest. Ebaõnnestunud päringu vastus näha Joonis 25. Kuigi reaalsuses sellist hulka päringuid korraga oodata ei ole, oli testide mõte kontrollida loodud rakenduse jõudlust ning leida punkt, kus rakendus enam päringuid vastu võtta ja töödelda ei suuda.



Joonis 25. PÄVIS2 koormustesti ebaõnnestunud päring

6 Kokkuvõte

Käesoleva diplomitöö käigus arendas töö autor välja uue Ohutuse infosüsteemi tulekahjujuhtumite haldusmooduli. Tagarakendus sai loodud OIS põhirakendusest eraldi, mille tulemusel on seda lihtsam hallata. Serveri poolel otsustas autor Java ja Spring Booti kasuks ning kasutajaliidese pool sai tehtud kasutades Angular 9 raamistikku. Valminud rakendus võtab vastu SOS tulekahjujuhtumite andmeid kasutades Apache ActiveMQ suhtlusteenust ning suhtlus PÄVIS2 ja OIS põhirakendusega toimub REST päringutega.

Valminud töö lihtsustab Päästeameti menetlejate tööd võimaldades saada ajakohast infot tulekahjujuhtumite kohta ning selle põhjal läbi viia lõplik juhtumite menetlemine. Saadud andmed on piisavad, et nende põhjal on võimalik erinevatel osapooltel teha väljavõtteid ja statistikat, mis annab asutusele sisendi juhtimisotsuste tegemiseks.

Tulekahjujuhtumite mooduli arendamine põhirakendusest eraldi parandab infosüsteemi hallatavust. Projektis valitud tehnoloogiate sarnasus juba varasemalt uuendatud põhirakenduse moodulite tehnoloogiatele vähendab infosüsteemi haldamise ja arendamisega tegeleva meeskonna uute tehnoloogiate õppimise koormust. Valitud uued tehnoloogiad on siiani ühed populaarsemad maailmas ja pidevas arengus, mis tähendab pikemaajalist toetust arendavate meeskondade poolt.

Diplomitöö raames tehtud töö skoop ei kata kogu projektis kavandatavat arendust. Töö käigus arendatu tuleks katta täiendavate automaattestidega. Lisafunktsionaalsustena on vaja välja arendada põhjalikum CSV väljavõte ning luua ATeS valehäirete teavituste süsteem. Täiendada on vaja ka serveri poolset sisendandmete validatsiooni.

Kasutatud kirjandus

[1] „What is AMQP and why is it used in RabbitMQ?,“ [Võrgumaterjal]. Available: https://www.cloudamqp.com/blog/what-is-amqp-and-why-is-it-used-in-rabbitmq.html. [Kasutatud 31 03 2021].

[2] „What is HTTP - W3Schools,“ [Võrgumaterjal]. Available: https://www.w3schools.com/whatis/whatis_http.asp. [Kasutatud 11 05 2020].

[3] „Java Message Service (JMS),“ [Võrgumaterjal]. Available: https://www.oracle.com/java/technologies/java-message-service.html. [Kasutatud 03 04 2021].

[4] A. Inc., „PDF Reference,“ [Võrgumaterjal]. Available: https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf. [Kasutatud 28 03 2020].

[5] „What is REST,“ [Võrgumaterjal]. Available: https://restfulapi.net/. [Kasutatud 03 30 2021].

[6] „User account and Authentication (UAA) server,“ Cloud foundry, [Võrgumaterjal]. Available: https://docs.cloudfoundry.org/concepts/architecture/uaa.html. [Kasutatud 13 09 2020].

[7] „XML Essentials - W3C - World Wide Web Consortium,“ [Võrgumaterjal]. Available: https://www.w3.org/standards/xml/core. [Kasutatud 01 03 2021].

[8] „Riigi infosüsteemi haldussüsteem,“ [Võrgumaterjal]. Available: https://www.riha.ee/Infosüsteemid/Vaata/pais. [Kasutatud 05 09 2020].

[9] C. Hoffman, „What is a CSV file, and how do I open it?,“ 17 04 2018. [Võrgumaterjal]. Available: https://www.howtogeek.com/348960/what-is-a-csv-file-and-how-do-i-open-it/. [Kasutatud 20 10 2020].

[10] „PostgreSQL: The World's Most Advanced Open Source Relational Database,“ The PostgreSQL Global Development Group, [Võrgumaterjal]. Available: https://www.postgresql.org/. [Kasutatud 20 03 2020].

[11] „DB-Engines Ranking - Trend of MySQL vs. Oracle vs. PostgreSQL Popularity,“ [Võrgumaterjal]. Available: https://db-engines.com/en/ranking_trend/system/MySQL%3BOracle%3BPostgreSQL. [Kasutatud 20 03 2020].

[12] „DB-Engines Ranking of Relational DBMS,“ [Võrgumaterjal]. Available: https://db-engines.com/en/ranking/relational+dbms. [Kasutatud 20 03 2020].

[13] „AngularJS — Superheroic JavaScript MVW Framework,“ Google, [Võrgumaterjal]. Available: https://angularjs.org/. [Kasutatud 30 03 2021].

[14] „Developer Guide: Introduction - AngularJS,“ [Võrgumaterjal]. Available: https://docs.angularjs.org/guide/introduction. [Kasutatud 30 03 2021].

[15] „Miscellaneous: Version Support Status - AngularJS,“ [Võrgumaterjal]. Available: https://docs.angularjs.org/guide/introduction. [Kasutatud 2021 03 30].

[16] „Comparing the most popular frontend JavaScript frameworks,“ [Võrgumaterjal]. Available: https://www.educative.io/blog/compare-popular-frontend-javascript-frameworks. [Kasutatud 30 03 2021].

[17] „Angular vs React vs Vue: Which Framework to Choose in 2021,“ [Võrgumaterjal]. Available: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#popularity. [Kasutatud 30 03 2021].

[18] „Choosing the Best Front-end Framework | Toptal,“ [Võrgumaterjal]. Available: https://www.toptal.com/javascript/choosing-best-front-end-framework. [Kasutatud 30 03 2021].

[19] „Introduction - RxJS,“ [Võrgumaterjal]. Available: https://rxjs-dev.firebaseapp.com/guide/overview. [Kasutatud 22 03 2020].

[20] „Bootstrap The most popular HTML, CSS, and JS library in the world.,“ [Võrgumaterjal]. Available: https://getbootstrap.com/. [Kasutatud 20 04 2020].

[21] „Aadressiandmete süsteemi andmeid sisaldav integreeritav aadressiotsingu kasutajaliides (In-ADS),“ [Võrgumaterjal]. Available: https://inaadress.maaamet.ee/inaadress/. [Kasutatud 12 01 2021].

[22] „Redis,“ [Võrgumaterjal]. Available: https://redis.io/. [Kasutatud 31 03 2021].

[23] „Spring Framework,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-framework. [Kasutatud 31 03 2021].

[24] „What is Spring Boot,“ [Võrgumaterjal]. Available: https://www.developer.com/java/ejb/what-is-spring-boot.html#:~:text=Spring%20has%20always%20favoured%20convention,for%20a%20specific%20application%20development.. [Kasutatud 31 03 2021].

[25] „Spring Boot,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-boot. [Kasutatud 31 03 2021].

[26] „Micronaut,“ 04 2019. [Võrgumaterjal]. Available: https://www.thoughtworks.com/radar/languages-and-frameworks/micronaut#:~:text=Micronaut%20is%20a%20JVM%20framework,using%20Java%2C%20Kotlin%20or%20Groovy.&text=Micronaut%20is%20a%20new%20JVM,footprint%20and%20short%20startup%20time.. [Kasutatud 12 03 2020].

[27] „Performance Comparison Between Spring Boot and Micronaut,“ 04 2019. [Võrgumaterjal]. Available: https://piotrminkowski.com/2019/04/09/performance-comparison-between-spring-boot-and-micronaut/. [Kasutatud 25 05 2020].

[28] „Spring Boot Reference Guide 2.1.3 RELEASE,“ [Võrgumaterjal]. Available: https://docs.spring.io/spring-boot/docs/2.1.3.RELEASE/reference/pdf/spring-boot-reference.pdf. [Kasutatud 31 03 2021].

[29] G. ROCHER, „MICRONAUT 1.0 RC1 AND THE POWER OF AHEAD-OF-TIME COMPILATION,“ 09 2018. [Võrgumaterjal]. Available: https://micronaut.io/2018/09/30/micronaut-1-0-rc1-and-the-power-of-ahead-of-time-compilation/. [Kasutatud 31 03 2021].

[30] „IntelliJ IDEA Capable and Ergonomic IDE for JVM,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/idea/. [Kasutatud 31 03 2021].

[31] „Spring Initializr,“ [Võrgumaterjal]. Available: https://start.spring.io/. [Kasutatud 05 03 2020].

[32] „Spring Data,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-data. [Kasutatud 31 03 2021].

[33] „ActiveMQ - The Apache Software Foundation!,“ [Võrgumaterjal]. Available: https://activemq.apache.org/. [Kasutatud 31 03 2021].

[34] „Messaging that just works — RabbitMQ,“ [Võrgumaterjal]. Available: https://www.rabbitmq.com/. [Kasutatud 02 04 2021].

[35] „FakeSMTP – FakeSMTP - Dummy SMTP server for developers,“ [Võrgumaterjal]. Available: http://nilhcem.com/FakeSMTP/. [Kasutatud 25 03 2020].

[36] „Cloud Foundry – Open Source Cloud Native Application Delivery,“ [Võrgumaterjal]. Available: https://www.cloudfoundry.org/. [Kasutatud 01 04 2021].

[37] „Docker: Empowering App Development for Developers,“ [Võrgumaterjal]. Available: https://www.docker.com/. [Kasutatud 02 04 2021].

[38] „Docker Desktop for Mac and Windows | Docker,“ [Võrgumaterjal]. Available: https://www.docker.com/products/docker-desktop. [Kasutatud 02 04 2021].

[39] „DockerHub redis,“ [Võrgumaterjal]. Available: https://hub.docker.com/_/redis. [Kasutatud 02 04 2021].

[40] „DockerHub webcenter/activemq,“ [Võrgumaterjal]. Available: https://hub.docker.com/r/webcenter/activemq. [Kasutatud 02 04 2021].

[41] „DockerHub scality/s3server,“ [Võrgumaterjal]. Available: https://hub.docker.com/r/scality/s3server. [Kasutatud 02 04 2021].

[42] „Flyway: Homepage,“ [Võrgumaterjal]. Available: https://flywaydb.org/. [Kasutatud 03 04 2021].

[43] „Spring Data JPA,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-data-jpa. [Kasutatud 03 04 2021].

[44] „Project Lombok,“ [Võrgumaterjal]. Available: https://projectlombok.org/. [Kasutatud 03 04 2021].

[45] „Faster Operations with the JSONB Data Type in PostgreSQL,“ [Võrgumaterjal]. Available: https://www.compose.com/articles/faster-operations-with-the-jsonb-data-type-in-postgresql/. [Kasutatud 03 04 2021].

[46] „vladmihalcea/hibernate-types: The Hibernate Types library- GitHub,“ [Võrgumaterjal]. Available: https://github.com/vladmihalcea/hibernate-types.

[47] „Using Spring JMS,“ [Võrgumaterjal]. Available: https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/jms.html. [Kasutatud 03 04 2021].

[48] baeldung, „Guide to JAXB,“ 17 08 2019. [Võrgumaterjal]. Available: https://www.baeldung.com/jaxb. [Kasutatud 03 04 2021].

[49] „Spring Security Basic Authentication,“ [Võrgumaterjal]. Available: https://www.baeldung.com/spring-security-basic-authentication. [Kasutatud 03 04 2021].

[50] „Working with Amazon S3,“ [Võrgumaterjal]. Available: https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/examples-s3.html. [Kasutatud 03 04 2021].

[51] „wkhtmltopdf,“ [Võrgumaterjal]. Available: https://wkhtmltopdf.org/. [Kasutatud 03 04 2021].

[52] „Template Engines for Spring,“ [Võrgumaterjal]. Available: https://www.baeldung.com/spring-template-engines. [Kasutatud 03 04 2021].

[53] „ExcelCSVPrinter - ostermiller.org,“ [Võrgumaterjal]. Available: https://ostermiller.org/utils/doc/com/Ostermiller/util/ExcelCSVPrinter.html. [Kasutatud 03 04 2021].

[54] „AMQP 0-9-1 Model Explained,“ [Võrgumaterjal]. Available: https://www.rabbitmq.com/tutorials/amqp-concepts.html. [Kasutatud 03 04 2021].

[55] „Hoonetulekahjudel päästetööga ärahoitud varakahju hindamise metoodika,“ [Võrgumaterjal]. Available: https://digiriiul.sisekaitse.ee/handle/123456789/2183. [Kasutatud 04 04 2021].

[56] „Spring Data Redis,“ [Võrgumaterjal]. Available: https://docs.spring.io/spring-data/data-redis/docs/current/reference/html/#reference. [Kasutatud 03 04 2021].

[57] „In-app navigation: routing to views,“ [Võrgumaterjal]. Available: https://angular.io/guide/router. [Kasutatud 04 04 2021].

[58] „Angular Authentication: Using Route Guards,“ [Võrgumaterjal]. Available: https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3. [Kasutatud 04 04 2021].

[59] „Anglar powered Bootstrap,“ [Võrgumaterjal]. Available: https://ng-bootstrap.github.io/#/components/modal/examples. [Kasutatud 04 04 2021].

[60] „Introduction to forms in Angular,“ [Võrgumaterjal]. Available: https://angular.io/guide/forms-overview. [Kasutatud 04 04 2021].

[61] „Angular – Lifecycle hooks,“ [Võrgumaterjal]. Available: https://angular.io/guide/lifecycle-hooks. [Kasutatud 05 04 2021].

[62] „RxJS - Introduction,“ [Võrgumaterjal]. Available: https://rxjs-dev.firebaseapp.com/guide/overview. [Kasutatud 05 04 2021].

[63] „JUnit 5,“ [Võrgumaterjal]. Available: https://junit.org/junit5/. [Kasutatud 05 04 2021].

[64] „Apache JMeter,“ [Võrgumaterjal]. Available: https://jmeter.apache.org/index.html. [Kasutatud 05 04 2021].

[65] „Micronaut Data,“ [Võrgumaterjal]. Available: https://micronaut-projects.github.io/micronaut-data/latest/guide/. [Kasutatud 31 03 2021].

[66] „Micronaut JMS,“ [Võrgumaterjal]. Available: https://micronaut-projects.github.io/micronaut-jms/snapshot/guide/index.html. [Kasutatud 31 03 2021].

[67] „MICRONAUT 2.3 RELEASED!,“ [Võrgumaterjal]. Available: https://micronaut.io/2021/01/22/micronaut-2-3-released/. [Kasutatud 31 03 2021].

[68] „Mockito framework site,“ [Võrgumaterjal]. Available: https://site.mockito.org/. [Kasutatud 05 04 2021].

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

Mina, Priit Rätsep

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Päästeameti ohutuse infosüsteemi tulekahjujuhtumite haldusmooduli arendus“, mille juhendaja on Meelis Antoi
   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.

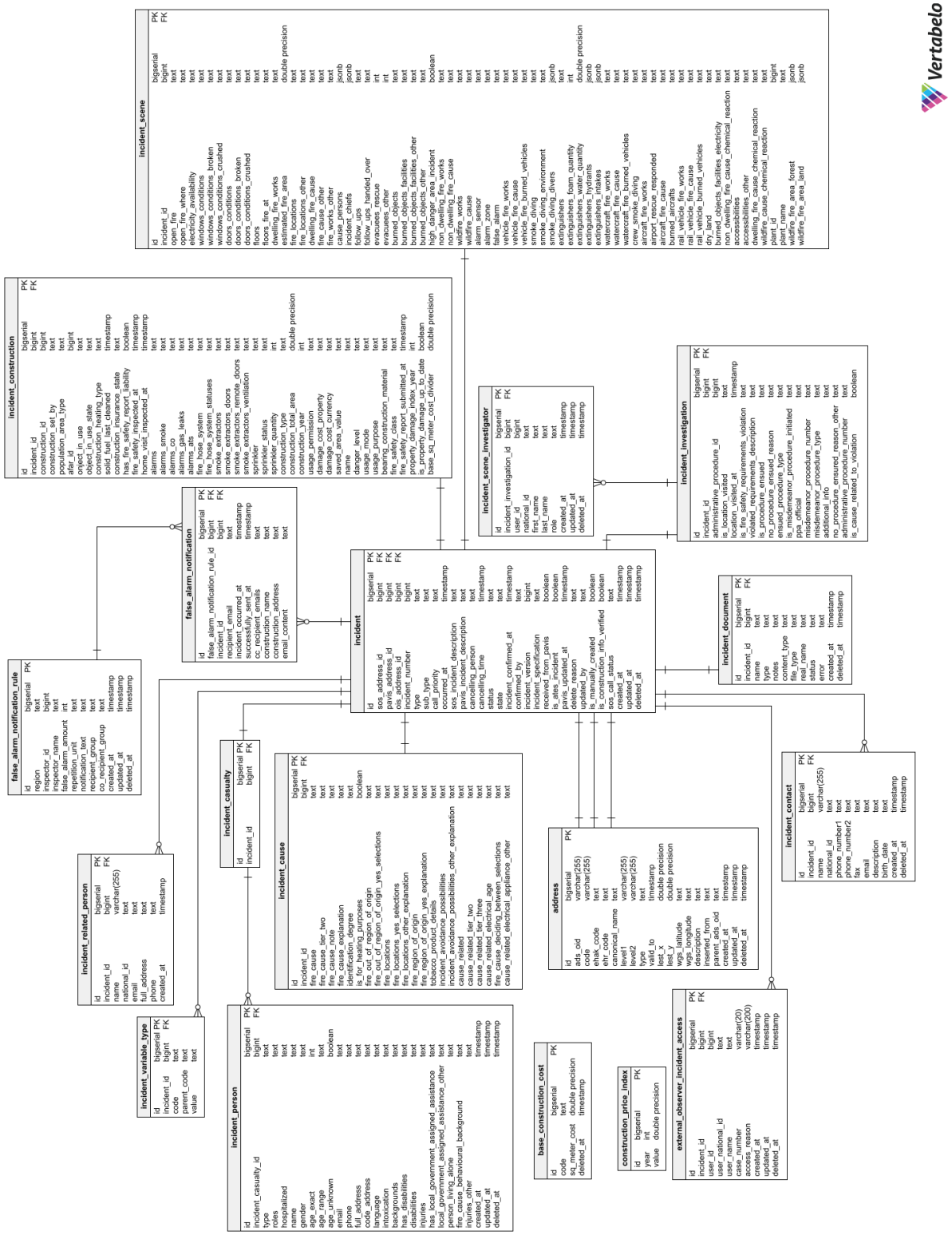
11.04.2021

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.

Lisa 2 – Olemite semantikad

Olem	Semantika
address	Tulekahjujuhtumi toimumise aadress. Eristatakse OIS, PÄVIS2 ja SOS aadresse.
base_construction_cost	Ehitise kasutusotstarvete järgi eristatud ühe ruutmeetri maksumus eurodes.
construction_price_index	Statistikaameti andmebaasist saadud ehitushinna koondindeksite muutused eelmise aastaga võrreldes aastate lõikes.
external_observer_incident_access	Välisvaatlejate juhtumite avamisi ja põhjuseid kajastav tabel.
false_alarm_notification	Teavitus valehäirete reegli alusel tuvastatud liigsete juhtumite ATS valehäirete kohta.
false_alarm_notification_rule	ATS valehäire reeglid.
incident	Tulekahjujuhtumi üldinfo.
incident_casualty	Tulekahjujuhtumis osalenud isikute vahetabel, hetkel pole erilist väärtus, aga on loodud tulevikuarenduste jaoks.
incident_cause	Tulekahjujuhtumi põhjuste detailinfo.
incident_construction	Juhtumiga seotud ehitise detailinfo.
incident_contact	Juhtumist teataja (helistaja) kontaktid.
incident_document	Tulekahjujuhtumile menetluse käigus lisatud dokumentide detailinfo.
incident_investigation	Tulekahjujuhtumi menetluse detailinfo.
incident_person	Juhtumis osalenud isikute detailinfo.
incident_related_person	Juhtumiga seotud isikute info.
incident_scene	Juhtumi sündmuskohaga seotud info.
incident_scene_investigator	Sündmuskohal viibija (menetlejana või vaatlejana
incident_variable_type	Tulekahjujuhtumi jaoks abitabel, kus on võimalik kajastada erinevate klassifikaatorite hierarhiate alusel väärtusi.

Lisa 3 – Olemi-suhte diagramm



Lisa 4 – Väljavõte *build.gradle* failist

```
dependencies {
    implementation("org.springframework.boot:spring-boot-starter-activemq")
    implementation("org.springframework.boot:spring-boot-starter-data-jpa")
    implementation("org.springframework.boot:spring-boot-starter-data-rest")
    implementation("org.springframework.boot:spring-boot-starter-security")
    implementation("org.springframework.boot:spring-boot-starter-web")
    implementation("org.springframework.boot:spring-boot-starter-thymeleaf")
    implementation("org.springframework.boot:spring-boot-starter-amqp")
    implementation("org.springframework.boot:spring-boot-starter-data-redis")
    implementation("org.flywaydb:flyway-core:${flywayVersion}")
    implementation("org.springframework:spring-oxm")
    implementation("org.springframework.retry:spring-retry")
    implementation("org.springframework.ws:spring-xml")
    implementation("commons-io:commons-io:2.6")
    implementation("org.osgeo:proj4j:0.1.0")
    implementation("org.apache.commons:commons-lang3:3.5")
    implementation("org.apache.commons:commons-math3:3.5")
    implementation("com.vladmihalcea:hibernate-types-52:2.9.5")
    implementation "com.amazonaws:aws-java-sdk-s3:1.11.690"
    implementation('org.ostermiller:utils:1.07.00')

    runtimeOnly("org.postgresql:postgresql:${postgreVersion}")
    compileOnly("org.projectlombok:lombok:${lombokVersion}")
    annotationProcessor("org.projectlombok:lombok:${lombokVersion}")
    annotationProcessor("org.springframework.boot:spring-boot-configuration-
    processor")

    testCompileOnly("org.projectlombok:lombok:${lombokVersion}")
    testAnnotationProcessor("org.projectlombok:lombok:${lombokVersion}")
    testImplementation("org.springframework.boot:spring-boot-starter-test") {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
    testImplementation "org.springframework.security:spring-security-test"
    testImplementation "junit:junit"
    testImplementation (group: 'org.mockito', name: 'mockito-core', version:
    '3.2.4') {
        exclude group: 'org.hamcrest'
    }
}

test {
    useJUnitPlatform()
}

def profiles = 'dev'
bootRun {
    jvmArgs = ["-Duser.timezone=Europe/Tallinn"]
    systemProperties System.properties
    args = ["--spring.profiles.active=" + profiles]
}
```

Lisa 5 – Väljavõte *application.yml* failist

```
server:
  port: 8085

spring:
  http:
    log-request-details: true
  datasource:
    driver-class-name: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5431/syndmus
    username: syndmus
    password:
  jpa:
    database-platform: org.hibernate.dialect.PostgreSQL10Dialect
    show-sql: false
    generate-ddl: false
    properties:
      hibernate:
        default_schema: "syndmus"
        format_sql: true
    hibernate:
      ddl-auto: validate
  flyway:
    locations: classpath:db/migration
  activemq:
    broker-url: tcp://localhost:61616
  servlet:
    multipart:
      max-file-size: '25MB'
      max-request-size: '25MB'
  rabbitmq:
    host: localhost
    port: 5675
    username: guest
    password:
    automatic-recovery-enabled: true
    network-recovery-interval: 30000
  redis:
    host: localhost
    port: 6379
    password:

mq:
  incident-mail-send: 'incident.mail-send'
  incident-exchange: 'incident-mail-send-exchange'
```

```
jms:
  queue:
    sos.jais.call.syndmus: OIS-V2.Syndmus.In

ois:
  service:
    url: http://localhost:8080

geo:
  service:
    url: https://replaced

pavis2:
  service:
    url: https://replaced
    username: 'PAVIS2'
    token:

logging:
  level:
    org:
      springframework:
        web: DEBUG

s3:
  client:
    accessKey: 'accessKey1'
    accessSecret:
    endpointUrl: 'http://localhost:8000'
    bucketName: 'incident'
    region: 'eu-west-1'

statistics:
  service:
    enabled: true
    url: 'https://andmed.stat.ee'
    connect-timeout: 10000
    read-timeout: 10000
    api-version: 'v1'
```

Lisa 6 – API päringud

Päring (/incident)	Meetod	Vastus	Kirjeldus
/list	GET	List<IncidentListItemDto>	Kõikide tulekahju-juhtumite otsing
/list/{constructionId}	GET	List<IncidentListItemDto>	Juhtumite otsing ehitise ID järgi
/page, IncidentSearch	POST	Page<IncidentListItemDto>	Juhtumite pagineeritud otsing
/export-csv, IncidentSearch	GET	HttpServletResponse	CSV alla laadimine
/{incidentId}	GET	IncidentMinimalDto	Juhtumi minimaalsete andmete päring
/create	POST	Long	Uue juhtumi loomine
/{incidentId}/reason/ {deleteReason}	DELETE		Juhtumi kustutamine
/{incidentId}/main	GET	IncidentMainDto	Üldiinfo päring
/{incidentId}/main, IncidentMainSave	PUT	IncidentMainDto	Üldinfo uuendamine
/{incidentId}/scene	GET	IncidentSceneDto	Sündmuskoha info päring
/{incidentId}/scene, IncidentSceneSave	PUT	IncidentSceneDto	Sündmuskoha info uuendamine
/{incidentId}/construction	GET	IncidentConstructionDto	Ehitise info päring
/{incidentId}/construction, IncidentConstructionSave	PUT	IncidentConstructionDto	Ehitise info uuendamine
/{incidentId}/construction/ {incidentConstructionId}/ bind-construction, IncidentConstructionSave	POST	IncidentConstructionDto	Ehitise sidumine tulekahjujuhtumiga
/{incidentId}/construction/ {incidentConstructionId}/ unbind-construction	DELETE	IncidentConstructionDto	Ehitise lahti sidumine tulekahju-juhtumist

Päring (/incident)	Meetod	Vastus	Kirjeldus
/{incidentId}/main/ {incidentMainId}/ person, IncidentRelatedPersonSave	POST	IncidentRelatedPerson	Juhtumiga seotud isiku lisamine
/{incidentId}/main/ {incidentMainId}/ person, IncidentRelatedPersonSave	PUT	IncidentRelatedPerson	Juhtumiga seotud isiku muutmine
/{incidentId}/main/ {incidentMainId}/ person/ {incidentRelatedPersonId	DELETE		Juhtumiga seotud isiku kustutamine
/{incidentId}/main/ {incidentMainId}/caller, CallerSave	POST	Caller	Helistaja kontaktide lisamine
/{incidentId}/main/ {incidentMainId}/caller, CallerSave	PUT	Caller	Helistaja kontaktide muutmine
/{incidentId}/main/ {incidentMainId}/caller/ {callerId}	DELETE		Helistaja kontaktide kustutamine
/{incidentId}/main/ {incidentMainId}/ oisAddress, OisAddressSave	POST	AddressDto	Juhtumi aadressi lisamine
/{incidentId}/investigation	GET	IncidentInvestigationDto	Menetluse info päring
/{incidentId}/investigation	POST	IncidentInvestigationDto	Menetluse info muutmine
/document/create, IncidentDocumentSave	POST	IncidentDocumentDto	Dokumendi lisamine
/document/{documentId}	GET	IncidentDocumentDto	Dokumendi info paring
/ document/delete/ {documentId}	DELETE		Dokumendi kustutamine
/document/download/ {documentId}	GET	HttpServletResponse	Dokumendi alla laadimine
/{incidentId/cause	GET	IncidentCauseDto	Põhjuste info paring

Päring (/incident)	**Meetod**	**Vastus**	**Kirjeldus**
/{incidentId/cause, IncidentCauseSave	PUT	IncidentCauseDto	Põhjuste info muutmine
/{incidentId/casualty	GET	IncidentCasualtyDto	Isikute info päring
/{incidentId/casualty, IncidentCasualtySave	PUT	IncidentCasualtyDto	Isikute info muutmine
/{id}/download-pdf, IncidentSearch	GET	ResponseEntity<byte[]>	Juhtumi PDF-I alla laadimine
/forward-incident	POST		Juhtumi edastamine e-kirjale
/{incidentId}/calculate-damages, CalculateProperty-DamagesCommand	POST	CalculatePropertyDamages-Response	Varakahju arvutamine
/{incidentId}/external-observer-access	GET	ExternalObserverIncident-AccessDto	Välisvaatleja eelmise juhtumi avamise info paring
/{incidentId}/external-observer-access, External-ObserverAccessCommand	POST	ExternalObserverIncident-AccessDto	Välisvaatleja esmakordne juhtumi avamine 24h jooksul

Lisa 7 – Dokumendi domeeniobjekti kood

```
@Data
@Entity
@Table(name = "incident_document")
@SQLDelete(sql = "UPDATE incident_document SET deleted_at = now() where id =
    ?")
@Where(clause = "deleted_at is NULL")
@EqualsAndHashCode(callSuper = true)
public class IncidentDocument extends BaseEntity {

    private String name;

    private String type;

    private String notes;

    private String contentType;

    private String fileType;

    private String realName;

    @Enumerated(EnumType.STRING)
    private UploadStatus status;

    private String error;

    @ManyToOne
    @JoinColumn(name = "incident_id")
    private Incident incident;

    @Column(name = "created_at")
    @Temporal(TemporalType.TIMESTAMP)
    private Date createdAt;

    @Column(name = "deleted_at")
    @Temporal(TemporalType.TIMESTAMP)
    private Date deletedAt;

    @PrePersist
    public void onCreate() {
        this.createdAt = new Date();
    }

    public enum UploadStatus {
        PENDING,
        CONFIRMED,
        FAILED
    }
}
```

Lisa 8 – Klassifikaatori objektide konverteerimise kood

```
@Converter(autoApply = false)
public class ClassifierConverter implements AttributeConverter<ClassifierDto,
String> {

    @Override
    public String convertToDatabaseColumn(ClassifierDto classifierDto) {
        if(classifierDto == null) return null;
        return classifierDto.getCode();
    }

    @Override
    public ClassifierDto convertToEntityAttribute(String code) {
       if(code == null)return null;
       if(code.equals(""))return null;
        ClassifierDto c = new ClassifierDto();
        c.setCode(code);
        return c;
    }
}
```

Lisa 9 – SOS sõnumi vastu võtmise lühendatud kood

```
public SosCallReceiver(){
   Jaxb2Marshaller jaxMarshaller = new Jaxb2Marshaller();
   jaxMarshaller.setClassesToBeBound(Valjakutse.class);
   this.marshaller = jaxMarshaller;
}

@JmsListener(destination="${jms.queue.sos.jais.call.syndmus}", concurrency =
"2-8")
public void receiveMessage(String message, @Header(value = "paaste", required
    = false) Boolean isRescueIncident) {
   if(isRescueIncident == null || !isRescueIncident) {
      logger.info("SOS-lt tuli sündmus mida ignoreeritakse kuna OIS võtab
          vastu ainult Pääste sündmused");
      return;
   }
   Valjakutse valjakutse = parseXml(message);
   if (valjakutse == null)
      return;
   StringBuilder loggerText = new StringBuilder("");
   try {
      Incident incident = importFromSos(valjakutse, loggerText);
      if (incident == null) {
         logger.warn("SOS-lt tuli Pääste sündmus. Väljakutse nr " +
             valjakutse.getValjakutseNr() + ". Sündmust ei salvesta järgmiste
             põhjusel(-tel): " + loggerText);
      }
   }catch (Exception e) {
      logger.error("SOS-lt tuli Pääste sündmus. Väljakutse nr " +
          valjakutse.getValjakutseNr() + ". Andmete impordimisel tekkis viga
          ja syndmus ei ole salvestatud. Põhjus: " + e.getCause());
   }
}

public Valjakutse parseXml(String xml) {
   try{
      JAXBElement<Valjakutse> jaxbElement = (JAXBElement<Valjakutse>)
      marshaller.unmarshal(new StringSource(xml));
      return jaxbElement.getValue();
   } catch(XmlMappingException e){
     logger.error("SOS-lt tuli Pääste sündmus. XML-i parsimisel tekkis
         jargmine viga: ", e);
   }
   return null;
}
```

```
public Incident importFromSos(Valjakutse valjakutse, StringBuilder
loggerText) throws Exception {

   return retryTemplate.execute(new RetryCallback<Incident, Exception>() {
      @Override
      public Incident doWithRetry(RetryContext context) throws Exception {
         String msg = "Attempting to import incident.";
         if (context.getRetryCount() > 0) {
            msg += " (retry #" + context.getRetryCount() + ")";
         }
         logger.info(msg);
         return doImportFromSos(valjakutse, loggerText);
      }
   });
}
```

Lisa 10 – PÄVIS2 saadetava tulekahjujuhtumi vastu võtmise põhimeetodi kood

```
public Incident createOrUpdateIncidentFromEvent(Event event) {
        Date incidentDeletedAt = incidentService.getIncidentDeletedAt(new
            IncidentNumber(event.getId()));
        if (incidentDeletedAt != null) {
            logger.info(" Sündmus on OISis varasemalt kustutatud. Sõnumit
                ignoreeritakse. ");
            return null;
        }
        Incident incident = incidentService.getIncident(new
        IncidentNumber(event.getId()));
        if (incident != null && incident.getIncidentConfirmedAt() != null) {
            logger.info(" Sündmus on OISis kinnitatud. Sõnumit
                ignoreeritakse. ");
            return null;
        }
        if (event.getState().equals("TRAINING")) {
            if (incident != null) {
                incidentService.deleteWithReason(incident,
                    DeleteReason.TRAINING);
                logger.info(" Sündmus on õppus. Sündmus kustutatakse. ");
                return null;
            } else {
                logger.info(" Sündmus on õppus. Sõnumit ignoreeritakse. ");
                return null;
            }
        }
        if (incident == null) {
            incident = new Incident(new IncidentNumber(event.getId()));
        }
        if (event.getType() != null && event.getType().equals("UNSPECIFIED"))
        {
          boolean eventHandled = handleUnspecifiedEvent(event);
          if (!eventHandled) {
              return null;
          }
        }


//        registered - sündmuse esmase registreerimise aeg
//        started - pääste valdkonna vaates sündmuse alguse aeg
        incident.setOccurredAt(event.getRegistered());
        incident.setIsAtesIncident(false);
        applyIncidentType(incident, event);
        resolvePavisAddress(incident, event);
        if (incident.getStatus() == null ||
```

```
    incident.getStatus().getValue().equals("")) {
        incident.setStatus(TYYPJUHTUM_OLEK_UUS);
    }
    applyIncidentState(incident, event);


    incident.setPavisIncidentDescription(
        event.getAdditionalDescription()
    );
    incident.setReceivedFromPavis(true);
    updateIncidentRelatedPersons(incident, event);
    incidentMainDataMapper.update(incident, event);
    resolveIncidentConfirmation(incident, event);
    ClassifierCodeValueEnum highestCallPriority =
        getHighestCallPriority(event, incident);
    if (highestCallPriority != null) {
        incident.setCallPriority(highestCallPriority.getValue());
    }
    incident = incidentService.save(incident, false);
    resolveIncidentTabs(incident, event);
    incident.setPavisUpdatedAt(new Date());
    incidentService.resolveIncidentConstruction(incident, "SYSTEM",
        PAVIS);

    return incident;
}
```

Lisa 11 – S3 andmehoidlaga suhtlemise kood

```
@Service
@Slf4j
@RequiredArgsConstructor
public class S3Service {
    private final S3Properties config;
    private AmazonS3 s3client;

    @PostConstruct
    private void initialize() {
        log.info("Starting S3 connection to server {} on region {}!",
            config.getEndpointUrl(), config.getRegion());
        AWSCredentials credentials = new
        BasicAWSCredentials(this.config.getAccessKey(),
            this.config.getSecretKey());
        this.s3client = AmazonS3ClientBuilder
                .standard()
                .withEndpointConfiguration(new
                 AwsClientBuilder.EndpointConfiguration(
                     config.getEndpointUrl(), config.getRegion()))
                .withPathStyleAccessEnabled(true)
                .withCredentials(new
                     AWSStaticCredentialsProvider(credentials))
                .build();

        if (!s3client.doesBucketExistV2(config.getBucketName())) {
            log.info("Creating new S3 bucket {}!", config.getBucketName());
            s3client.createBucket(config.getBucketName());
        }
    }

    public S3Object getS3Object(String pathName, Long fileId) {
        return s3client.getObject(config.getBucketName(), pathName + "/" +
            fileId);
    }

    public InputStream getFileContent(String pathName, Long fileId) {
        log.info("Requesting file from S3 {}", pathName + "/" + fileId);
        S3Object s3 = getS3Object(pathName, fileId);
        return s3.getObjectContent();
    }

    public void deleteFile(String pathName, Long fileId) {
        log.info("Deleting file from S3 {}", pathName + "/" + fileId);
        s3client.deleteObject(config.getBucketName(), pathName + "/" +
            fileId);
    }
}
```

Lisa 12 – CSV faili genereerimise kood

```
@Service
@Log4j2
public class CsvService {
    @Autowired
    OisApiService oisApiService;

    @Autowired
    IncidentRepository incidentRepository;

    public byte[] getCsvContent(IncidentSearch searchParams) throws
            UnsupportedEncodingException {
        StringWriter stream = new StringWriter();
        ExcelCSVPrinter csvPrinter = new ExcelCSVPrinter(stream);
        csvPrinter.changeDelimiter(';');
        csvPrinter.setAlwaysQuote(true);
        csvPrinter.println(getCsvArray(searchParams));

        return stream.toString().getBytes("CP1252");
    }

    public String[][] getCsvArray(IncidentSearch searchParams) {
        IncidentSearchSpec spec = new IncidentSearchSpec(searchParams,
            oisApiService);
        List<Incident> list = incidentRepository.findAll(spec);
        String[][] csvArray = new String[list.size() + 1][8];

        SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");

        String[] header = {"Sündmuse number", "Sündmuse liik", "Toimumise
                aeg", "Aadress", "Koordinaat X", "Koordinaat Y", "Aadressi
              täpsustus", "Staatus"};
        csvArray[0] = header;
        for (int i = 0; i < list.size(); i++) {
            Incident incident = list.get(i);
            String[] row = new String[8];
            row[0] = incident.getIncidentNumber().getValue().toString();

            ClassifierDto typeClassifierDto = null;
            if (incident.getType() != null) {
                typeClassifierDto =
                oisApiService.getClassifier(incident.getType().getValue());
            }
            if (typeClassifierDto != null) {
                row[1] = typeClassifierDto.getLabel();
            } else {
                row[1] = incident.getType() != null ?
                incident.getType().getValue() : "";
            }

            row[2] = incident.getOccurredAt() != null ?
```

```
                sdf.format(incident.getOccurredAt()) : "";

            Address oisAddress = incident.getOisAddress();
            Address sosAddress = incident.getSosAddress();
            Address pavisAddress = incident.getPavisAddress();
            Address currentAddress = null;
            if (oisAddress != null) {
                currentAddress = oisAddress;
            } else {
                if (pavisAddress != null && sosAddress == null) {
                    currentAddress = pavisAddress;
                } else if (pavisAddress == null && sosAddress != null) {
                    currentAddress = sosAddress;
                } else if (pavisAddress != null && sosAddress != null) {
                    currentAddress = pavisAddress.getUpdatedAt().compareTo(
                        sosAddress.getUpdatedAt()) > 0 ? pavisAddress :
                        sosAddress;
                }
            }

            if (currentAddress != null) {
                row[3] = currentAddress.getCanonicalName() != null ?
                    currentAddress.getCanonicalName() : "";
                row[4] = currentAddress.getLestX() != null ?
                    currentAddress.getLestX().toString() : "";
                row[5] = currentAddress.getLestY() != null ?
                    currentAddress.getLestY().toString() : "";
                row[6] = currentAddress.getDescription() != null ?
                    currentAddress.getDescription() : "";
            } else {
                row[3] = row[4] = row[5] = row[6] = "";
            }

            ClassifierDto stateClassifierDto = null;
            if (incident.getState() != null) {
                stateClassifierDto = oisApiService.getClassifier(
                    incident.getState().getValue());
            }
            if (stateClassifierDto != null) {
                row[7] = stateClassifierDto.getLabel();
            } else {
                row[7] = incident.getState() != null ?
                    incident.getState().getValue() : "";
            }

            csvArray[i + 1] = row;
        }
        return csvArray;
    }
}
```

Lisa 13 – RabbitMQ konfiguratsiooni kood

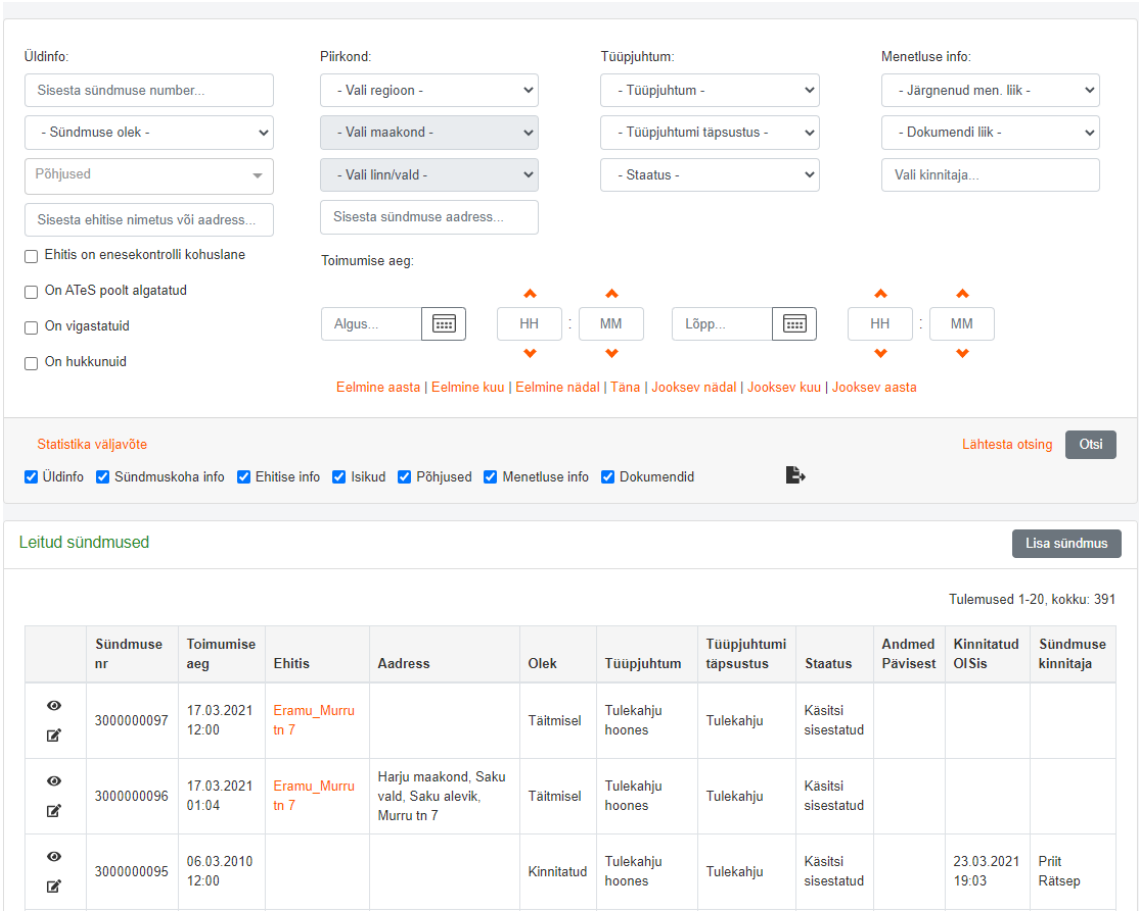
```
@Configuration
public class RabbitConfiguration {

    @Value("${mq.incident-mail-send:'incident.mail-send'}")
    String queueName;

    @Value("${mq.incident-exchange:'incident.exchange'}")
    String exchange;

    @Bean
    Queue queue() {
        return new Queue(queueName);
    }

    @Bean
    DirectExchange exchange() {
        return new DirectExchange(exchange);
    }

    @Bean
    Binding binding(Queue queue, DirectExchange exchange) {
        return BindingBuilder.bind(queue).to(exchange).with(queueName);
    }

    @Bean
    public RabbitTemplate rabbitTemplate(
            final ConnectionFactory connectionFactory
    ) {
        final var rabbitTemplate = new RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(
            producerJackson2MessageConverter()
        );
        return rabbitTemplate;
    }

    @Bean
    public Jackson2JsonMessageConverter producerJackson2MessageConverter() {
        return new Jackson2JsonMessageConverter();
    }
}
```

Lisa 14 – Redis teenuse kood

```
@Transactional
@Service
public class ExternalObserverIncidentAccessService {

    @Autowired
    UserService userService;
    @Autowired
    private RedisTemplate<String, String> redisTemplate;
    @Autowired
    ExternalObserverIncidentAccessRepository
    externalObserverIncidentAccessRepository;

    private String appPrefix = "incident_";

    public ExternalObserverIncidentAccessDto create(Incident incident,
        ExternalObserverAccessCommand command) throws JsonProcessingException
    {
        OisUserDto currentUser = userService.getCurrentUser();

        ExternalObserverIncidentAccess externalObserverIncidentAccess = new
            ExternalObserverIncidentAccess();
        externalObserverIncidentAccess.setIncident(incident);
        externalObserverIncidentAccess.setCaseNumber(
            command.getCaseNumber());
        externalObserverIncidentAccess.setAccessReason(
            command.getAccessReason());
        externalObserverIncidentAccess.setUserId(currentUser.getId());
        externalObserverIncidentAccess.setUserNationalId(
            currentUser.getNationalId());
        externalObserverIncidentAccess.setUserName(currentUser.getName());

        externalObserverIncidentAccess =
        externalObserverIncidentAccessRepository.save(
            externalObserverIncidentAccess);

        ExternalObserverIncidentAccessDto dto = new

        ExternalObserverIncidentAccessDto(externalObserverIncidentAccess);

        ObjectMapper objectMapper = new ObjectMapper();

        ValueOperations<String, String> values = redisTemplate.opsForValue();
        values.set(appPrefix + incident.getId() + "_" +
        currentUser.getNationalId(), objectMapper.writeValueAsString(dto),
```

```
        24, TimeUnit.HOURS);

        dto.setIncidentNumber(incident.getIncidentNumber().getValue());
        return dto;
    }

    public ExternalObserverIncidentAccessDto getPreviousAccess(Incident
        incident) throws JsonProcessingException {
        OisUserDto currentUser = userService.getCurrentUser();
        String key = appPrefix + incident.getId() + "_" +
        currentUser.getNationalId();

        if(!isAccessValid(key)){
            return null;
        }

        ValueOperations<String, String> values = redisTemplate.opsForValue();
        String str = values.get(key);
        if (str == null) {
            return null;
        }
        ObjectMapper objectMapper = new ObjectMapper();

        ExternalObserverIncidentAccessDto dto = objectMapper.readValue(str,
            ExternalObserverIncidentAccessDto.class);
        dto.setIncidentNumber(incident.getIncidentNumber().getValue());
        return dto;
    }

    public Long getTtl(String key) {
        return redisTemplate.getExpire(key, TimeUnit.SECONDS);
    }

    private boolean isAccessValid(String key){
        Long ttl = getTtl(key);
        if(ttl <= 0){
            this.deleteAccess(key);
            return false;
        }
        return true;
    }

    public void deleteAccess(String key) {
        redisTemplate.delete(key);
    }
}
```

Lisa 15 – Juhtumi otsingu nimekirja vaade



Lisa 16 – Klientrakenduse marsruutimise detailide kood

```
const routes: Routes = [
  {
    path: 'incident',
    component: IncidentListComponent
  },
  {
    path: 'incident/:incidentId/:subTab',
    children: [
      {
        path: '', redirectTo: 'edit', pathMatch: 'full'
      },
      {
        path: 'edit',
        component: IncidentComponent,
        canActivate: [IncidentEditGuard]
      },
      {
        path: 'view',
        component: IncidentComponent,
        canActivate: [IncidentViewGuard],
        data: {
              readOnly: true
            }
      }
    ],
    resolve: {
      incident: IncidentResolver
    },
    runGuardsAndResolvers: 'always'
  },
];
@NgModule({
  imports: [
    RouterModule.forChild(routes)
  ],
  exports: [RouterModule],
  providers: [
    IncidentResolver,
    IncidentEditGuard,
    IncidentViewGuard
  ]
})
export class IncidentRoutingModule { }
```

Lisa 17 – Tulekahjujuhtumi detailvaate muutmise Guard

```
@Injectable({
  providedIn: 'root'
})
export class IncidentEditGuard implements CanActivate {

  constructor(private auth: AuthService,
              private route: ActivatedRoute,
              private router: Router) {
  }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Promise<boolean> {

    return new Promise<boolean>((resolve) => {
      let currentUser = this.auth.currentUser;
      if (!currentUser || !IncidentHelper.isUserAllowedToEditIncident(current
          User)) {
        this.router.navigate(['incidents/incident', route.params.incidentId,
           route.params.subTab, 'view'], {relativeTo: this.route})
          .then(() => {});
        resolve(false);
        return;
      } else {
        resolve(true);
      }
    });
  }
}
```

Lisa 18 – Helistaja kontaktide vormi HTML kood

```
<form [formGroup]="callerForm" novalidate autocomplete="off"
  (ngSubmit)="onSubmit()">
  <div class="modal-header">
    <h4 class="modal-title">Helistaja kontakt</h4>
    <button type="button" class="close"
      data-dismiss="modal" (click)="activeModal.dismiss()">&times;</button>
  </div>

  <div class="modal-body">
    <div class="form-group form-row">
        <label for="name" class="col-form-label required">Nimi:</label>
        <input id="name" type="text" class="form-
          control" formControlName="name"
          [ngClass]="{'is-invalid': submitted
            && callerForm.controls['name'].invalid}"/>
    </div>
    <div class="form-group form-row">
      <label for="phoneNumber1" class="col-form-label required">
        Telefon:
      </label>
      <input id="phoneNumber1" type="text" class="form-
        control" formControlName="phoneNumber1"
        [ngClass]="{'is-invalid': submitted
          && callerForm.controls['phoneNumber1'].invalid}"/>
    </div>
  </div>

  <div class="modal-footer">
    <button type="button" class="btn btn-default"
      (click)="activeModal.dismiss()">Katkesta</button>
    <button type="submit" class="btn btn-success">Salvesta</button>
  </div>
</form>
```

Lisa 19 – Tulekahjujuhtumite teenuse klassi lühendatud kood

```
@Injectable({
  providedIn: 'root'
})
export class IncidentService {
  private baseUrl: string;
  public pageQueryCache: any = {};

  constructor(private http: HttpClient,
              private config: ConfigService) {
    this.baseUrl = config.get('incidentServiceUrl') + '/incident';
  }

  public page(params: any): Observable<any> {
    return this.http.post<any>(this.baseUrl + '/page', params);
  }

  public exportCsv(params: any): Observable<HttpResponse<Blob>> {
    const url = this.baseUrl + '/export-csv';
    return this.http.get(url, {
      params: params,
      observe: 'response',
      responseType: 'blob'
    });
  }

  public downloadPdf(incident: Incident, params: any):
    Observable<HttpResponse<Blob>> {
    const url = this.baseUrl + '/' + incident.id + '/download-pdf/';
    return this.http.get(url, {
      params,
      observe: 'response',
      responseType: 'blob'
    });
  }
}
```

Lisa 20 – SOS sõnumi vastu võtmise automaattesti testpäringu sisu

```
<valjakutse>
    <valjakutseNr>2003020007</valjakutseNr>
    <olek>VALJAKUTSE_OLEK_TOOS_RESSURSITA</olek>
    <piirkond>PIIRKOND_POHJA</piirkond>
    <paaste>
        <syndmusLiikKood>17A</syndmusLiikKood>
        <syndmusLiik>TULEKAHJU HOONES</syndmusLiik>
        <prioriteet>PRIORITEET_PAASTE_1</prioriteet>
        <kysimustik>
            <kysimusVastus>
                <kysimusKood>1705</kysimusKood>
                <vastusKood>5352</vastusKood>
                <kysimusTekst>Millega on tegemist?</kysimusTekst>
                <vastusTekst>
                    rajatise tulekahju (lisainfosse, mis rajatis)
                </vastusTekst>
                <kysimusVastusKood>362129</kysimusVastusKood>
            </kysimusVastus>
        </kysimustik>
        <soovituslikPrioriteet>PRIORITEET_PAASTE_1</soovituslikPrioriteet>
        <valdkonnaOlek>VALJAKUTSE_OLEK_TOOS_RESSURSITA</valdkonnaOlek>
    </paaste>
    <registreerimiseAeg>2020-03-02T15:54:34</registreerimiseAeg>
    <lisainfo>See osdaff</lisainfo>
    <kontakt>
        <nimi>Priit Rätsep</nimi>
        <telefon>042234232</telefon>
        <email></email>
    </kontakt>
    <sosAadressKuva>
        Pargi tn 6-21, Saku alevik, Saku vald, Harju mk
    </sosAadressKuva>
    <sosSyndmusLiikKuva>TULEKAHJU HOONES - 17A</sosSyndmusLiikKuva>
    <sosPrioriteetKuva>1</sosPrioriteetKuva>
    <valjakutseVersioon>0</valjakutseVersioon>
    <saadaKinnitus>false</saadaKinnitus>
    <valjakutseAeg>2020-03-02T15:54:34</valjakutseAeg>
    <muutmiseAeg>2020-03-02T15:54:34</muutmiseAeg>
    <patsientArv>0</patsientArv>
    <muutja>Priit Rätsep</muutja>
    <sisestaja>
        <isikukood>38805110255</isikukood>
```

```
        <nimiKuva>Priit Rätsep</nimiKuva>
        <polStruktYksus />
    </sisestaja>
    <asukoht>
        <koodaadress>37719736100000722000020MR9TIA0000</koodaadress>
        <version>0</version>
        <taisaadress>
            Harju maakond, Saku vald, Saku alevik, Pargi tn 6-21
        </taisaadress>
        <lisainfo>Keldris</lisainfo>
        <x>6573647.82</x>
        <y>537771.5</y>
        <ehakKood>37719736100000000000000000000000</ehakKood>
        <tekkAeg>2020-03-02T15:54:34</tekkAeg>
        <maantee>
            <nimetus />
            <nr />
            <km />
        </maantee>
        <aadress>
            <tanav>Pargi tn</tanav>
            <maja>6</maja>
            <korter>21</korter>
        </aadress>
        <objekt>
            <taisnimetus />
            <nimetus />
        </objekt>
    </asukoht>
</valjakutse>
```

Lisa 21 – SOS sõnumi vastu võtmise testi kood

```
@RunWith(SpringRunner.class)
@SpringBootTest(classes = SyndmusApplicationTests.class)
@ActiveProfiles(profiles = "test")
@SqlGroup({
        @Sql(executionPhase = Sql.ExecutionPhase.BEFORE_TEST_METHOD, scripts
          = "classpath:scripts/insert-incident.sql"),
        @Sql(executionPhase = Sql.ExecutionPhase.AFTER_TEST_METHOD, scripts =
         "classpath:scripts/clear-data.sql")
})
public class SosCallReceiverTest {

    private static final Logger logger =
        LoggerFactory.getLogger(SosCallReceiverTest.class);

    @Autowired
    private IncidentRepository incidentRepository;

    @Autowired
    private JmsTemplate jmsTemplate;

    @Value("${jms.queue.sos.jais.call.syndmus}")
    private String destination;

    @Test
    public void receiveMessage() throws Exception {
        Long incidentNumber = 2003020007L;

        String xml = readFileToString(ResourceUtils.getFile(
            "classpath:docs/valjakutsePOIParing.xml"
        ), StandardCharsets.UTF_8);
        logger.info(
            "SOS-st tulnud sõnum: " + System.lineSeparator() + "{}", xml
        );



        this.jmsTemplate.send(destination, session -> {
            TextMessage message = session.createTextMessage(xml);
            message.setBooleanProperty("paaste", true);
            return message;
        });

        Thread.sleep(3000);

        Incident incident = incidentRepository.findByIncidentNumber(new
```

```
            IncidentNumber(incidentNumber));
        IncidentScene incidentScene = incident.getIncidentScene();

        Assert.assertNotNull(incident);
        Assert.assertEquals(Long.valueOf(2003020007),
            incident.getIncidentNumber().getValue());
        Assert.assertEquals("TYYPJUHTUM_LIIK_TULEKAHJU_HOONES",
            incident.getType().getValue());
        Assert.assertNull(incident.getSubType());
        Assert.assertEquals("INCIDENT_PRIORITY_1",
            incident.getCallPriority());
        Assert.assertEquals("Harju maakond, Saku vald, Saku alevik, Pargi tn
            6-21", incident.getSosAddress().getCanonicalName());
        Assert.assertEquals(6573647.82, incident.getSosAddress().getLestX(),
            0.001);
        Assert.assertEquals(537771.5, incident.getSosAddress().getLestY(),
            0.001);
        Assert.assertEquals("377197361000000000000000000000000",
            incident.getSosAddress().getEhakCode());
        Assert.assertEquals("37719736100000722000020MR9TIA0000",
            incident.getSosAddress().getCode());
        Assert.assertEquals(Long.valueOf(1), incidentScene.getId());
        Assert.assertEquals(Long.valueOf(2),
            incidentScene.getIncident().getId());
    }
}
```