TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ekaterina Afanasjeva 206723IADB

Korduvate klientide leidmise lahendus

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

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: Ekaterina Afanasjeva

25.03.2023

Annotatsioon

Antud töö eesmärk on lahendada laevatootmise infosüsteemi töötajate topeltkirjete tekkimise probleem. Töös on analüüsitud probleemi põhjuseid ning kaalutud selle lahendamise võimalusi. Töös on valitud ja rakendatud üks lahendus.

Laeva tootmise töötajate hulgas tekivad topeltkirjed, sest nad kõik on seotud töövõtuahelas ühe tööandjaga. Kuna rakenduses ei ole piiranguid, mitu korda saab tööandjat ahelasse lisada, siis igale tööandja kirjele tekivad töötajad isegi kui tööandja ahelas juba esineb. Seega võib üks ja sama töötaja olla laeva tootmisele lisatud mitu korda, mis põhjustab vigu kliendi- ja serverirakenduses.

Suur osa tööst on pühendatud äriprotsessi loogika funktsionaalsuse analüüsimisele, mida tuleb muuta. Diplomitöö hõlmab projekti ja probleemi lahendamisel kasutatud tehnoloogiaid ja tööriistu ning lõpus on esitatud lahenduse loomine koos koodi näidetega ja valitud lahenduse test tõhususe analüüsimiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab 59 lehekülge, 7 peatükki, 29 joonist ning 2 tabelit.

Abstract

Duplicate Users Finding Solution

The aim of this thesis is to solve the problem of duplicate records in the information system for ship production. In this work, an analysis of the reasons for the problem was conducted, as well as options for its solution were considered. One of them was selected and implemented.

Duplicates occur in employee records within the production of the selected ship, as all of them are associated with different records of the same employer in the employment chain. There are no restrictions on the number of records of the same employer in the same chain, and employees are added specifically to them. The same employee can be added to the production of one ship several times, which causes bugs on the client application side and errors in the system on the server application side.

A large part of the work is devoted to analyzing the business logic in the functionality that needs to be changed. The thesis covers the technologies and tools used in the project and the solution, and in the final part of the work, the creation of a solution with code examples is considered and test is conducted to analyze the effectiveness of the selected solution.

The thesis is in Estonian and contains 59 pages of text, 7 chapters, 29 figures, 2 tables.

Lühendite ja mõistete sõnastik

DTO	*Data Transfer Object* - andmeedastusobjekt
ERD	*Entity Relationship Diagram* - Olemi-suhte diagramm
FIE	Füüsilisest isikust ettevõtja
HTTP	*Hypertext Transfer Protocol* – edastusprotokoll
JSON	*Javascript Object Notation* - Javascriptil põhinev andmevahetusvorming
KÜ	Klassifikatsiooniühing
LT	Laevade tootmine
PTV	Peatöövõtja
REST	*Representational State Transfer* – veebiteenusega suhtlemise liidese arhitektuuri stiil
SQL	*Structured Query Language* - Struktuurpäringukeel
Two-way binding	Kahepoolne andmete sidumine
URL	*Uniform Resource Locator* – veebiaadress
VIEW	*Virtual table whose contents are defined by a query* - virtuaalne tabel, mille sisu on määratletud päringuga

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänasel päeval on projektide arendamisel palju erinevaid lähenemise viise. Kõige levinum neist on agiilne arendusmudel. Sellisel juhul ei avalda meeskond kogu funktsionaalsust korraga, vaid töö jaguneb väikesteks rühmadeks, mis täidetakse järjestikku ja tarnitakse kliendile. Seega toimub hindamine ja töö analüüs pidevalt ning valmis funktsionaalsus jõuab peaaegu kohe testimisele ja vigade parandamine on väga kiire. [1]

Ühelt poolt on see üsna mugav, kuna kõik projektis on paindlik ja toimub samal ajal. Kui arendamise käigus tekivad küsimused ja täiendavad selgitused, saadakse neile vastused väga kiiresti. Samuti saavad arendajad mõjutada, kuidas funktsionaalsus lõpuks töötab, kuna kõik muudatused ja kohandused on võimalik kooskõlastada.

Teisest küljest on pidevate muutuste jälgimine väga keeruline. Mõne nädala jooksul võib sama funktsioon muutuda mitu korda ja iga kord võib seda muuta erinev arendaja. Lõpptulemusena on kood kirjutatud mitme inimese poolt, mistõttu probleemide otsimine ja lahendamine võtab rohkem aega kui peaks. Funktsionaalsus ei tundu kunagi valmis ja nõuab pärast iga väikest muudatust pidevat testimist, kuna isegi väikese koodi muudatuse tegemine võib mõjutada kogu funktsionaalsuse tööd. See viib olukorrani, kus sama ülesanne liigub pidevalt analüüsi, arenduse ja testimise vahel. Probleeme tekib koodi kirjutamise, testimise ja analüüsi kiirusega.

Käesolevas lõputöös vaadatakse ja lahendatakse üks selline ülesanne. Alguses täielikult analüüsitud ja tehtud funktsionaalsus, mis seejärel läks testimisele. Hiljem tuli sellele teha väike täiendus ja ülesanne delegeeriti teisele arendajale. Arendamise käigus selgus, et funktsionaalsus ei tööta täpselt nii nagu klient ootas ning andmebaasi skeem ei sobi ülesande lahendamiseks. Täielikult valmis funktsionaalsus tuleb uuesti analüüsida ning seda tuleb praktiliselt nullist uuesti kirjutada võttes arvesse kliendi ootusi. Lahenduse variante oli mitu ning kõik need on selles lõputöös läbi vaadatud. Üks lahenduse variant on valitud ja rakendatud.

Kuna funktsionaalsus nõuab muudatusi andmebaasis ning sellega seotud valikuga tekkis raskusi, otsustati kasutada teist lähenemisviisi, kus andmeid koguti mitmest tabelist ühte. Tulemuse tõhususe analüüsimiseks testiti andmete päringu kiirust ning analüüsiti andmete mahtu. Tänu sellele saab näha, kuidas funktsionaalsus töötaks ideaalsetes tingimustes, kus see oleks täielikult uute nõuete kohaselt ümber ehitatud. See aitab ka mõista, kas lõpuks vastu võetud lahendus jääb kiiruse ja mugavuse poolest alla.

Põhiprojekt on tööjõu korraldamise infosüsteem LT-sel (laeva tootmine). Iga laev registreeritakse süsteemis objektiga ning erinevad ettevõtted ja FIE-d (Füüsilisest isikust ettevõtjad) osalevad tootmisprotsessis. Tööandjal on töötajate nimekiri ning mõned neist võivad LT-le lisanduda. Süsteem võimaldab ka laadida teavet tootmiskohas käimise kohta, et hiljem nende andmete põhjal näidata statistikat töötajate ja tööandjate viibimise aja kohta. Infosüsteem võimaldab aru saada, kes täpselt töötab tootmiskohtadel, kas sellel inimesel oli sel hetkel ligipääs ja millise tööandja töötaja ta on.

Konfidentsiaalsuse leping kliendiga seab teatud piirangud andmetele, mis võivad selles töös kirjeldatud olla. Rakenduse olemus on muudetud, kliendi nimi ei ole välja toodud, testimise ajal kasutatakse fiktiivseid isikuandmeid ning mõne ülesande sõnastust on muudetud ja mahtu vähendatud. Siiski jäävad probleemi kirjeldus ja selle lahendamise meetodid samaks.

2 Probleemi püstitus

Rakenduse põhifunktsioonid on järgmised:

- Laevade loomine ja registreerimine objektidena
- Tööandja lisamine süsteemi ja LT-le
- Firma esindajana töötaja lisamine süsteemi ja LT-le
- Pääsuvahendi nimekirja haldamine LT-l
- Töötajate pääsu haldamine LT-l

Peaaegu igal asjal on oma olek ja lisaparameetrid, mis tähendab, et tööprotsessi käigus luuakse palju erinevaid õiguste kombinatsioone ja piiranguid kasutajate võimalustele.

2.1 Töövõtuahela ja töötaja ülevaade

Tööandjad lisatakse LT-le puustruktuurina koos juurega, st loomisel määratakse laeval PTV (Peatöövõtja), kes lisab allhankijad ja igal allhankijal on teostatav tööde liik. See on niinimetatud TA(Töövõtuahel). Allhankija võib omakorda lisada enda alla teisi allhankijaid ja nii edasi. Rakenduses TA pikkuse piiranguid pole. Samuti pole piiranguid, mis näitaksid, mitu korda sama tööandja võib ahelas ilmuda. Seega võib sama tööandja olla mitme seosega sama laevaga, erinedes ainult TA asukohast ja tööde liigist. [2]

Kui funktsionaalsus oli alles loomisel, siis TA edasine kasutamine ei olnud veel analüüsitud. Seega oli vastuvõetav, et laeval oli täiesti sõltumatuid allhankija kirjeid. Edasi hakkas ilmuma funktsionaalsus, mis võimaldas tööandja nimel LT-le töötaja lisada ja märkida, kas ta võib LT-l osaleda või mitte. Kui töötaja LT-le lisati, siis oli andmebaasis otseühendus tema tööandja allhankija kirjega. Kui tööandjal on ainult üks allhankija kirje, siis lisatakse töötajad üksnes sellele ainsale olemasolevale kirjele. Kui töötaja pääsu muudetakse, muudetakse andmeid ainult ühes

kirjes ja kõik toimib õigesti. Kui tööandjal on kaks või rohkem kirjet, siis saab ta sama töötaja LT-le mitu korda lisada. Kõik need kirjed on ainulaadsed ja igal neist saab töötaja pääsu eraldi muuta. Seda ei peetud funktsionaalsuse analüüsimisel ja andmebaasi loomisel veaks. Pigem eeldati alguses, et kaks kirjet on täiesti normaalne, sest allhankijal on kaks eraldi tööde liiki. Seega võib üks inimene olla LT-le lisatud kui tootmise tugipostide ettevalmistuse insener ja teine kui korpuspuhastuse süsteemi tehnik. [3]

Joonis 1. Tööandja ilmub töövõtuahelas kaks korda

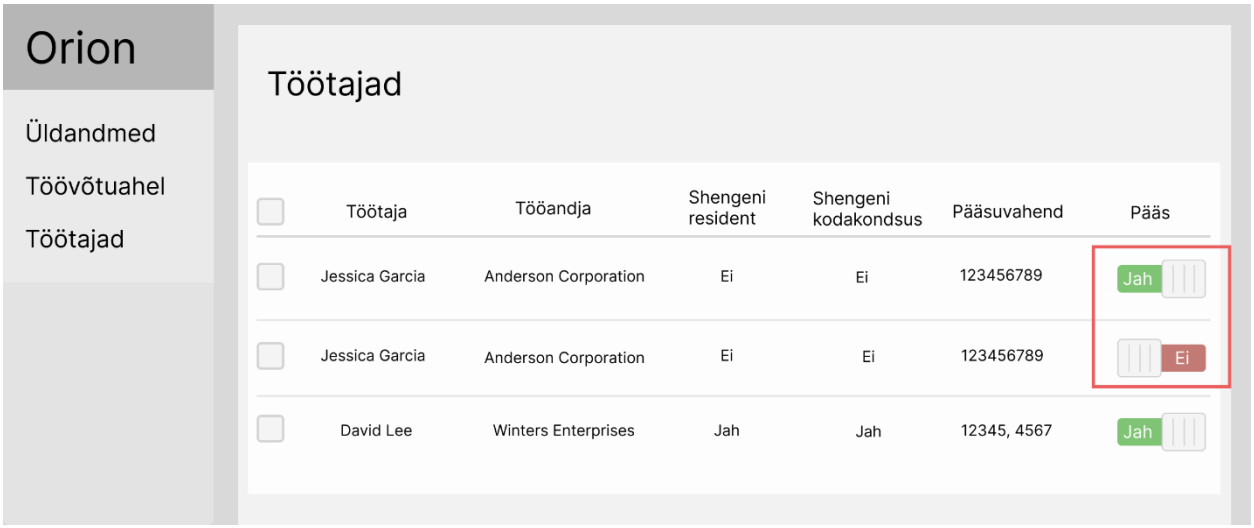
Pärast funktsionaalsuse demonstreerimist kliendile selgus, et süsteem ei toimi ootuspäraselt. Pole tähtsust, mitu korda tööandja ahelasse lisatakse, tal on ikkagi ainult üks töötaja ja tal on kas pääs tootmises osalemisele või mitte.

2.2 Andmete sünkroniseerimine

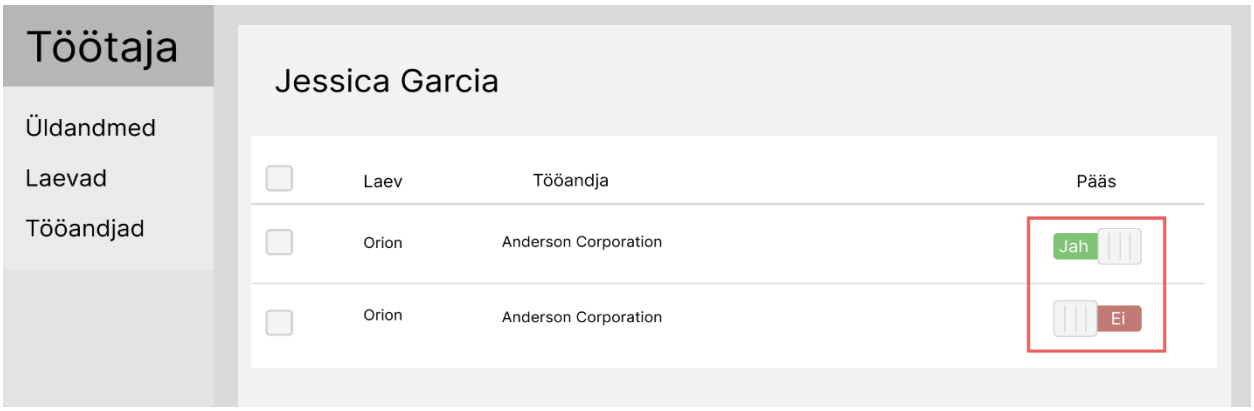
Kui tööandjal on ainult üks töötaja, kes saab olla laevaga seostud vaid üks kord, siis olemasolev funktsionaalsus töötab põhimõtteliselt valesti. Erinevused algavad just seal, kus üks kirje ütleb, et töötajal on pääs tootmisele, samas kui teine ütleb, et tal pole. Lõpptulemusena pole kellelegi selge, millised andmed on tõesed ja miks ühest ja samast tööandjast on mitu kirjet ühe ja sama inimese kohta. On võimalik hõlpsasti muuta pääsu ainult ühe kirje jaoks või lihtsalt mitte lisada töötajat iga allhankija kirje jaoks. Seega on andmed kas erinevad või puudu ning töötaja staatus LT-l pole kokkuvõttes arusaadav.

2.3 Vead veebilehel

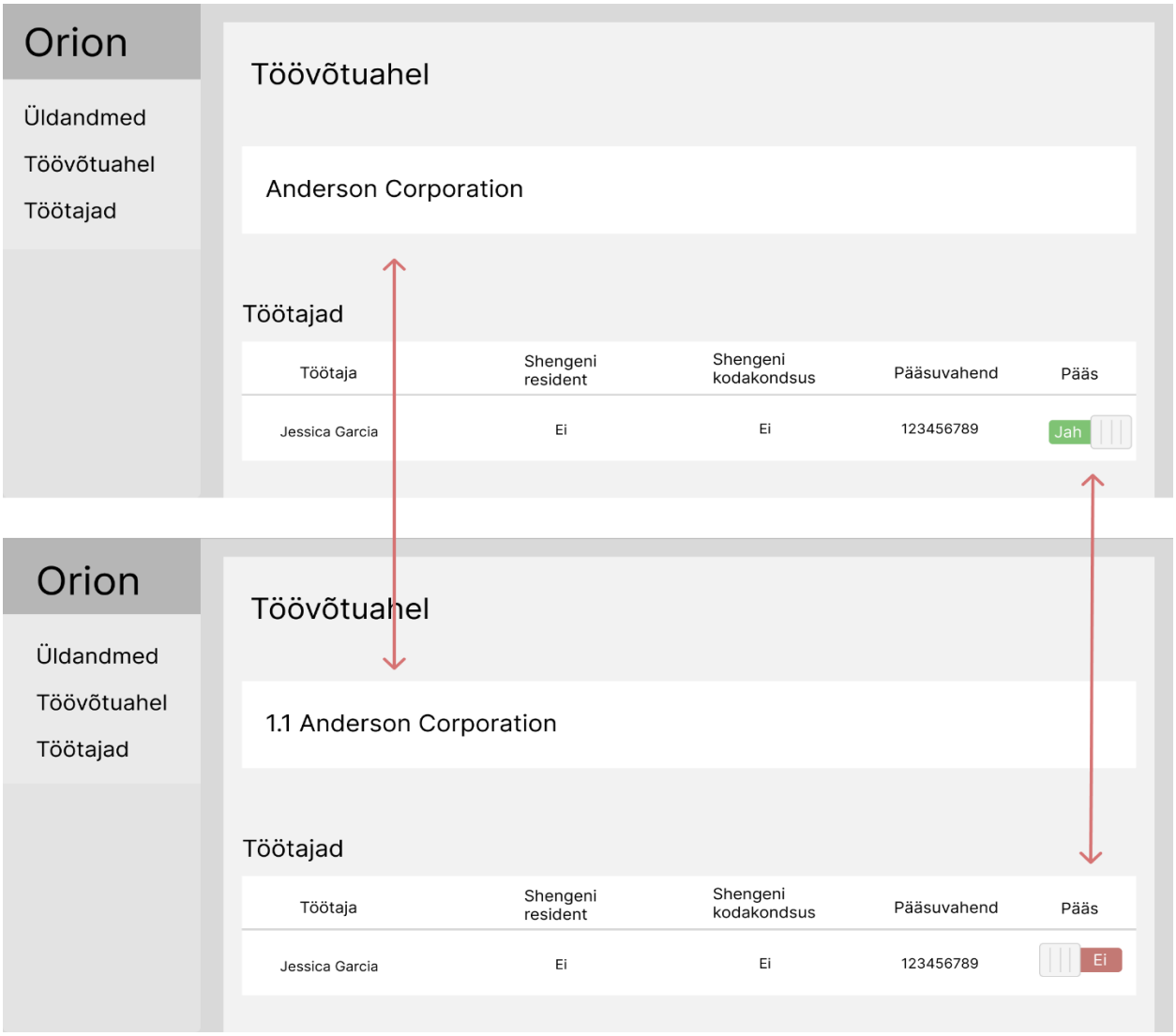
Andmete erinevused on kõige paremini nähtavad lehekülgedel, kus on LT-l olevate töötajate nimekiri ja nende praegused pääsud või lehekülgedel, mis näitavad ühe töötaja jaoks laevade nimekirja ja nende pääse. Tööandja näeb sama töötajat LT-l kaks korda, samas kui töötaja näeb endal kahte identset laeva, millest ühel on juurdepääs olemas, teisel aga mitte. Selliste vigade tõttu on funktsionaalsuse kasutamine ebamugav. Kasutajad ei tea, kuidas duplikaadid tekkisid ja neid ei saa eemaldada - saab ainult muuta staatust. Kui selliste andmete põhjal saab teavet LT külastamise kohta, siis hiljem saab tööandja teate, et tema töötaja oli tootmisel, kuigi tal sinna juurdepääsu pole. Kasutajale pole selge, millisest kahest või enamast töötajast räägitakse.



Joonis 2. Vead töötaja andmete lehel. Sama laev ilmub kaks korda



Joonis 3. Töötajate nimekiri LT-l. Sama töötaja ilmub kaks korda



Joonis 4. Vead allhankija lehel. Sama töötaja erineva pääsuga LT-le

2.4 Planeeritav andmete maht

Infosüsteemi peaks mahtuma suur hulk andmeid. Ühe aasta jooksul luuakse süsteemis umbes 1000 laeva. Üks LT kestab pool aastat. Seega luuakse täiesti nullist igal poolaastal 500 laeva. Iga laeva loomise protsessis osaleb umbes 20 allhankijat. Võib eeldada, et igal laeval on keskmiselt 20 kirjet, kus tööandjad ilmuvad ainult üks kord. Ühel LT alal võib olla üks pääsusüsteem, kuhu saab sisse kuni 4 erineva pääsuvahendiga. Kui iga pääsuvahend kuulub ainult ühele töötajale kogu ehituse ajal, siis on kokku kirjeid töötajate kohta LT-l 80. Igal tööpäeval lähevad töötajad tööle ja lahkuvad sealt mitu korda. Sisenemised ja väljumised registreeritakse ning saadetakse süsteemi. LT-sel olevate töötajate andmeid võib olla poolaasta jooksul umbes 40000. Aasta jooksul see arv kahekordistub.

3 Analüüs

Ülesanne nõuab analüüsi ja valiku tegemist, kuidas lahendus ellu viia ja põhjust, miks selline lahenduse valik on kõige efektiivsem. Selles lõputöö osas on esitatud funktsionaalsed ja mittefunktsionaalsed nõuded, kirjeldatud metoodikat ja funktsionaalsuse dokumentatsiooni ülevaadet. Samuti on loetletud kasutatavad tehnoloogiad ning on üksikasjalikult uuritud kõiki ülesande lahendamise võimalusi ja valitud üks neist edasiseks rakendamiseks.

3.1 Nõuded

Funktsionaalsed nõuded vastutavad selle eest, mida funktsionaalsus peab tegema õige toimimise tagamiseks. Sellel konkreetsel juhul peaks funktsionaalsus kas kõrvaldama võimaluse andmebaasi luua duplikaate või looma need ilma kasutaja teadmata ning täitma kõik töötaja staatuse arvutused iseseisvalt. Mõlemad variandid viivad üldise nõudeni: lõplik loend töötajatest ja nende pääsudest LT-le peab olema kasutajale arusaadavas vormis. Samal ajal ei tohiks süsteem põhjustada ootamatuid vigu ega kaotada töötajaid, kui üks tööandja kirje sulgub. Seni, kuni tööandjal on aktiivne kirje, peab ta nägema oma töötajaid LT-l ja neil peab olema võimalus ühe nupuvajutusega nende pääsu muuta. Sama nõue kehtib ka töötaja jaoks: nimekiri laevadest, mille LT-le ta on lisatud, peab olema arusaadav. Andmed kas rühmitatakse iga laeva jaoks või ilmub iga laev loendis ainult üks kord. [4]

Mittefunktsionaalsed nõuded viitavad sellele, mida süsteemis tuleb arvestada kvaliteetse töö tagamiseks. Funktsionaalsus peab suutma kiiresti töödelda suurt hulka kirjeid andmebaasis (lõigus 2.4 on toodud eeldatav andmevõimsus). Lehe laadimise aeg teabega ei tohiks kasutajale ebamugavust põhjustada. Sama kehtib ka olukorras, kus laevale lisatakse uus töötaja. Kui mitu tööandjat hakkavad korraga töötajatele juurdepääsu sulgema, siis ei tohiks see võtta palju aega. Koodi kvaliteedi seisukohalt on ka piiranguid - see peaks olema võimalikult arusaadav, et tulevikus leitud vigu oleks lihtne leida ja parandada. [4]

3.2 Tehnoloogiad

Antud osas on loetletud rakenduse arendamisel kasutatud tehnoloogiad ja töö kirjutamisel kasutatud vahendid. Iga tehnoloogia jaoks on välja toodud eelised, mis põhjustasid selle valiku. Mõnel juhul ei kaalutud analooge, kuna tehnoloogia oli juba tuttav ja sobib täielikult töö tegemiseks. Programmide osas on välja toodud mitte ainult töö praktilise osa teostamiseks vajalikud tööriistad, vaid ka tulemuste testimiseks ja töö loogilise skeemi ning demo disaini kujundamiseks vajalikud vahendid.

3.2.1 Raamistikud

Selles rakenduses keskendutakse just nendele tehnoloogiatele, mis on kliendile ja tellijale kõige paremini tuttavad. Firmale soovitatud serverirakenduse raamistik on Spring ning kliendirakenduste jaoks on Angular ja React. Firma, mis arendab valitud rakendust spetsialiseerub nendele tehnoloogiatele ja seega ei pea arendajad pidevalt vahetama tehnoloogiaid kui nad töötavad mitmel projektil korraga.

Populaarseks valikuks veebirakenduste serverirakenduse osa arendamiseks on Spring ja ASP.NET. Spring kasutab Java keelt, samas kui ASP.NET kasutab C#. Springi peamised eelised seisnevad selles, et see on avatud lähtekoodiga ja võimaldab importida ainult neid komponente, mis on rakenduse jaoks vajalikud, samas kui ASP.NET on võrdlemisi mahukas. [5] [6]

Angular toetab *Two-way bindingut* (Kahepoolne andmete sidumine), mis võimaldab mudelit ja selle kuvamist hoida sünkroonis. Reactis tuleb muutujate olekut käsitsi uuendada, mis muudab koodi tervikuna keerulisemaks. Angular pakub rohkem funktsioone kui React. Seega on paljudel Angulari rakendustel sarnane struktuur, mis võimaldab vajadusel lahendusi teistest projektidest laenata. React kasutab põhifunktsioonide jaoks kolmanda osapoole raamatukogusid, näiteks marsruutimine, mis annab rohkem valikuid, kuid erinevates projektides võib struktuur olla täiesti erinev. Lõpuks valiti raamistikuks Angular, kuna see sisaldab juba kõike vajalikku projekti alustamiseks ning see kiirendab tööd. [7]

3.2.2 Andmebaas

Algselt oli PostgreSQL klientide poolt nõutud andmebaaside haldamissüsteem. Klientidel oli juba valmis serverite ja andmebaaside süsteem ning kõik nad kasutasid just PostgreSQLi ja Tomcati.

PostgreSQLil on lihtsam SQL (*Structured Query Language* - Struktuurpäringukeel) süntaks kui näiteks Oracle'il, samal ajal kui tal on avatud lähtekood ning projekti raames saab luua ainulaadseid meetodeid. PostgreSQL jääb Oracle'ile alla ametliku dokumentatsiooni, usaldusväärsuse ja jõudluse osas, kuid ta on täiesti tasuta ja võimaldab kohandamist. PostgreSQL, Oracle, Microsoft SQL, MySQL toetavad SQL standardit, mis võimaldab kirjutada skripte, kasutades põhilisi SQLkäske, näiteks SELECT, CREATE, UPDATE, DROP. Kõik need süsteemid toetavad ka JSON (*Javascript Object Notation* - Javascriptil põhinev andmevahetusvorming) andmetüüpi, mida kasutatakse projektis palju ja lahendab probleemi, mida käsitletakse selles lõputöös. Võib järeldada, et PostgreSQL valiti kliendi jaoks juba tuttava tehnoloogia tõttu ning neil on oma spetsialistid, kes oskavad sellega töötada ja on sellega harjunud. [8] [9] [10] [11] [12]

3.2.3 Rakendusserver

Rakenduse jaoks olid peamised serverivalikud Tomcat, Jetty, GlassFish ja WildFly. Kõik need võivad töötada Java rakendustega. Rakendus peab toetama suurt arvu samaaegseid päringuid, näiteks tootmiskohtade külastatavuse teabe laadimine perioodi lõpus. Suure hulga samaaegsete päringutega saab Jetty paremini hakkama kui Tomcat kui mõlemad eristuvad just päringute töötlemise kiiruse poolest. Tomcat on vanem ja sellel on rohkem dokumentatsiooni, kolmanda osapoole pistikprogramme ja suurem kasutajate baas. Jetty on aga paindlikum ja seda on lihtsam rakendusse integreerida. GlassFish pole nii populaarne kui Tomcat või Jetty, selle seadistamine ja häälestamine on keerulisem. GlassFish töötleb päringuid aeglasemalt ja tal on vähem pistikprogramme kui Tomcatil. GlassFish sobib suurepäraselt Java ettevõtte taseme rakendustele, kuid valitud süsteem ei tohiks olla nii mahukas ja võimsam server pole vajalik. WildFly on sarnane GlassFishiga seadistamise raskuse ja sellega, et see sobib paremini ettevõtte tasemele. WildFly on paindlik ja võimaldab valida, milliseid funktsioone sisse lülitada ja milliseid välja jätta, kuid rakendusele pole vaja suurema jõudlusega serverit. Lõpuks on rakenduse realiseerimiseks vaja kergeid ja kiireid servereid, mis saavad töötada Javaga. Valik tehti Tomcati kasuks, kuna algsetel arendusetappidel oli palju aega serveri installimiseks ja seadistamiseks ning tulevikus pakub see rohkem võimalusi kui Jetty. [13] [14] [15]

3.2.4 Programmid

Töös on loogilised skeemid ja ERD (*Entity Relationship Diagram* - Olemi-suhte diagramm) andmebaasi mudelid, mille loomisel kasutati veebivahendit Lucidchart. Selle tööriista liides on arusaadav ja mugav ning sellega saab väga kiiresti luua diagramme, lohistada plokke ja joonistada nooli komponentide vahel. Rakendusel on tasuline tellimus, mis eemaldab piirangud skeemides olevate komponentide arvule, kuid selles töös pole ühtegi skeemi, mis vajaks rohkem kui 60 komponenti, seega kasutatakse tasuta versiooni ja selle võimalusi. [16]

Selleks, et luua lehekülje kujunduse prototüüpe ja näidata komponente, kus andmed kuvatakse valesti, kasutati Figmat. Figma võimaldab luua kujundeid ja kontuure, salvestada need valmis komponentidena ning seejärel kokku panna neist valmis pildi nagu konstruktoris. [17]

Testide läbiviimiseks on vaja luua 2 testandmebaasi erinevate ERD mudeliga. Projekti arendamisel kasutatakse Docker Desktopi, mis võimaldab igal arendajal omada eraldi testandmebaasi. Sellisel juhul üks arendaja ei mõjuta teise andmebaasi. Projekti konfiguratsiooni alusel luuakse pilt ja seejärel luuakse konteiner koos andmebaasiga. Selle töö raames kasutatakse 2 täiendavat konteinerit ühest peamisest valmis pildist, mida kasutatakse projekti arendamisel. Esimene konteiner on muutmata andmebaas täiendava vaate tabeliga. Teine konteiner on ümber kujundatud andmebaasi skeem, mis lahendab dubleeritud kirjed kohe andmebaasi tasemel. Kahe teineteisest sõltumatu konteineri kasutamise tõttu saab andmebaasi suurusi kontrollida kirjete arvu järgi. [18]

Rakenduse töötamiseks ja praktilise ning testi osa täitmiseks kasutatakse IntelliJ IDEA'd. Hetkel on see parim Java koodiga töötamise vahend, millega autor on kokku puutunud. Tööriista kasutamiseks on vaja litsentsi, mida pakub tööandja. Seda kasutatakse nii serverirakenduses kui ka kliendirakenduses, kuna see integreerub hästi Angulari, Springi ja Gradle'ga, mida projektis kasutatakse. Tööriist pakub mitmeid funktsioone, mis kiirendavad koodi kirjutamise ja õige vastuse leidmise protsessi. IntelliJ IDEA võimaldab kiiresti leida vigu ja puudusi koodis ning neid parandada. Lisaks toetab IntelliJ IDEA failide muutmise ajalugu, mis võimaldab kiiresti tagasi minna vajaliku koodi versioonini ilma *commitideta*. Loomulikult on sellel tööriistal analooge nagu Eclipse ja Visual Studio Code, kuid nende pakutavad võimalused ja kasutusmugavust ei saa võrrelda IntelliJ IDEAga. [19] [20] [21] [22]

3.3 Metoodika ja funktsionaalsuse dokumentatsioon

Ülesannete koostamisel ja nende lahenduste analüüsimisel tuleb kasutada õiget lähenemist. Meetod võimaldab saavutada kõige edukama lõpptulemuse. Iga suurema ülesande jaoks on kasulik koostada dokumentatsioon, mis võimaldab samm-sammult uurida kõiki süsteemi tegevusi igas olukorras. Selle funktsiooni eelmise versiooni jaoks on juba olemas dokumentatsioon. Meetodi töö põhimõte seisneb olemasoleva dokumentatsiooni täiustamises ja uue funktsionaalsuse koostamises koostöös analüütikutega. Sel viisil kooskõlastatakse süsteemi kõik töödetailid kohe, mis aitab tulevikus vigu ja süsteemi parandamist vältida.

Süsteemi põhiülesanne seisneb selles, et tööandja saaks oma töötaja LT-le lisada ja tema juurdepääsu kontrollida. Pääsu saab igal ajal sulgeda, kuid objekti-töötaja seos jääb siiski alles. Siiski pole see ainus viis, kuidas töötaja andmed LT-l võivad ilmuda või muutuda.

See funktsionaalsus mõjutab töötaja teavet LT-l:

- Tööandja töövõtuahelasse lisamine
- Allhankija staatuse muutmine
- Tööandja üldnimekirja lisamine
- Tööandja tootmisele lisamine
- Töötaja pääsu muutmine
- Töötaja staatuse muutmine
- Töötajate seadistuse muutmine (automaatselt töötaja tootmisele lisamine)

Just siin tekib LT-l samast töötajast erinevaid versioone. Kasutaja ei lisa alati käsitsi topeltversiooni, mõned süsteemid on automatiseeritud.

Funktsioonaalsus, mis võimaldab näha teavet töötaja pääsu LT-le:

- Töötaja detailandmete vaatamine

- Töötajate nimekiri LT-l
- Allhankija deailandmete vaatamine

Siin näeb kasutaja, et andmed võivad olla puudulikud või need ei klapi üksteisega. Ülesande lahendamine tähendab, et kõik meetodid, mis on seotud dubleeritud töötajatega, parandatakse.

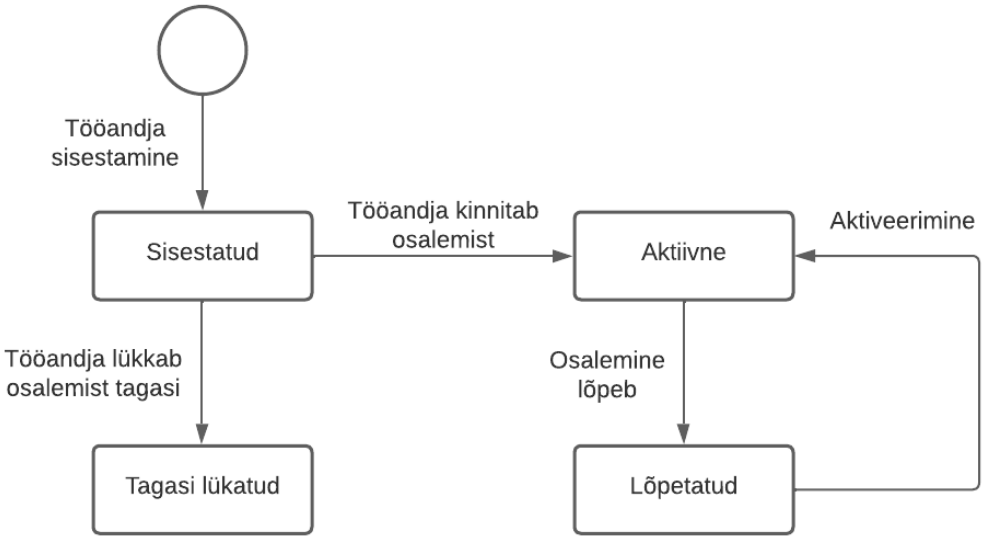
Järgmisena vaadatakse põhjalikult läbi praegune versioon andmebaasist ning kõik meetodid, mis loovad ja näitavad valesid andmeid.

Lõputöö raames on koostatud loogilised skeemid lihtsustatud funktsionaalsuse alusel. Selleks uuriti vanade rakenduste koodi ja dokumentatsiooni testjuhtumeid. Need on koostatud ka võimalike probleemilahenduste jaoks nii, et süsteemi loogika muutuse erinevust on mugavam arutada.

3.3.1 Andmebaas

Andmebaasi versioonide mudelid asuvad Lisas 4.

Allhankijal on 4 staatust. Kui allhankija on just lisatud, siis kirje olekut nimetatakse "Sisestatud". Kui tööandja lisatakse ahelasse, saab ta teavituse informatsiooniga siseneda süsteemi ja kinnitada oma osalemist LT-se töövõtuahelas. Kui tööandja oma osaluse kinnitab, siis muutub kirje olek ahelas aktiivseks. Kui tööandja keeldub, siis kirje lakkab olemast kehtiv ja ta märgitakse tagasilükatuks. Ahelasse lisatud tööandja ja PTV võivad firma osaluse LT-s lõpetada. Sel hetkel muutub kirje olek "Lõpetatud". Samuti saab lõpetatud kirje uuesti aktiveerida. See on tehtud juhuks kui keegi sulgeb kirje kogemata või kui töös oli pikemaid pause, näiteks mitu kuud.



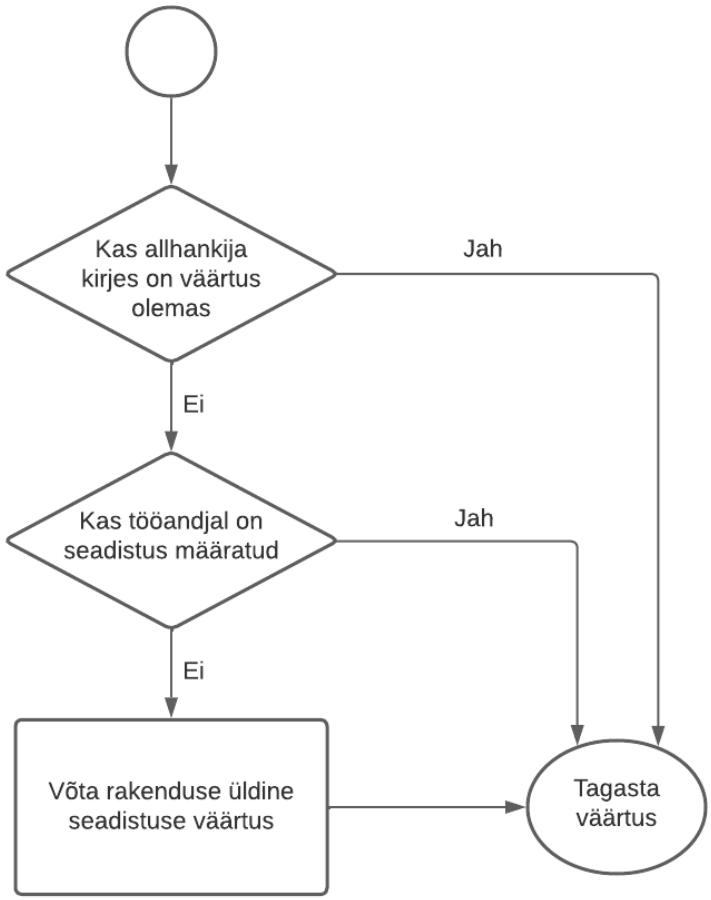
Joonis 5. Allhankija staatused

Lisaks tööandjatele LT-s osalevad kliendid ja KÜ (klassifikatsiooniühing) esindajad (laevaga seotud isikud). Nende tööks on kontrollida ja kinnitada tootmisdokumentatsiooni sobivust ning üle vaadata ja testida materjale ning seadmeid. Tööandjal, klientidel ja KÜ-l võivad olla oma ahelad. Kui kõik tööandjad on ühendatud ühe ahelaga, mis algab PTV-st, siis igal lisatud kliendil võib olla oma eraldi ahel. Sama kehtib ka KÜ kohta. Ahelate arv ühe laeva raames pole piiratud, kuid klient ei saa oma töötajaid LT-le lisada samal ajal kui teised allhankijad saavad.

Igal töötajal on isiklik tõend, samuti võivad igal tööandjal olla pääsuvahendid, mis sobivad konkreetsele LT-le juurdepääsuks. Tööandja võib ajutiselt anda oma töötajale pääsuvahendi. Kõik need dokumendid on eraldi tabelisse märgitud ja jagatud gruppidesse. Kui tööandja otsustab dokumendi ühelt töötajalt teisele üle anda, määratakse vanale kirjele lõppkuupäev ja uue omanikuga loodakse andmete duplikaat. Antud juhul, kus konkreetses ülesandes on vaja lisada ja muuta pääsuvahendite omanikke, oli vastav funktsionaalsus juba loodud ning seda eraldi ei arutleta. Neid tabelitest saadud andmeid kasutatakse funktsioonis, mis näitab töötajate nimekirja laeval, kus ühes veerus loetletakse töötaja aktiivseid kaarte.

Iga *employer* ja *employment_chain* kirjetel on seadistus (*employees_access_allowed*), mis näitab, kas uusi töötajaid saab automaatselt lisada uuele LT-le. Uute töötajate automaatne lisamine toimub

kas siis, kui tööandja kinnitab oma osalemise LT-s või kui tööandja lisab uue töötaja. Tabelitel on prioriteet, *employment_chain* väärtus on olulisem kui *employer* väärtus, nii et tööandjal võib olla üldine reegel, et uusi töötajaid ei lisata automaatselt, kuid mõne laeva jaoks tehakse erandeid. Nende muutujate väärtused ei ole kohustuslikud ja kui automaatse lisamise kohta pole mingit teavet, võetakse väärtus rakenduse üldistest süsteemi seadetest. Vaikimisi on see "ei".

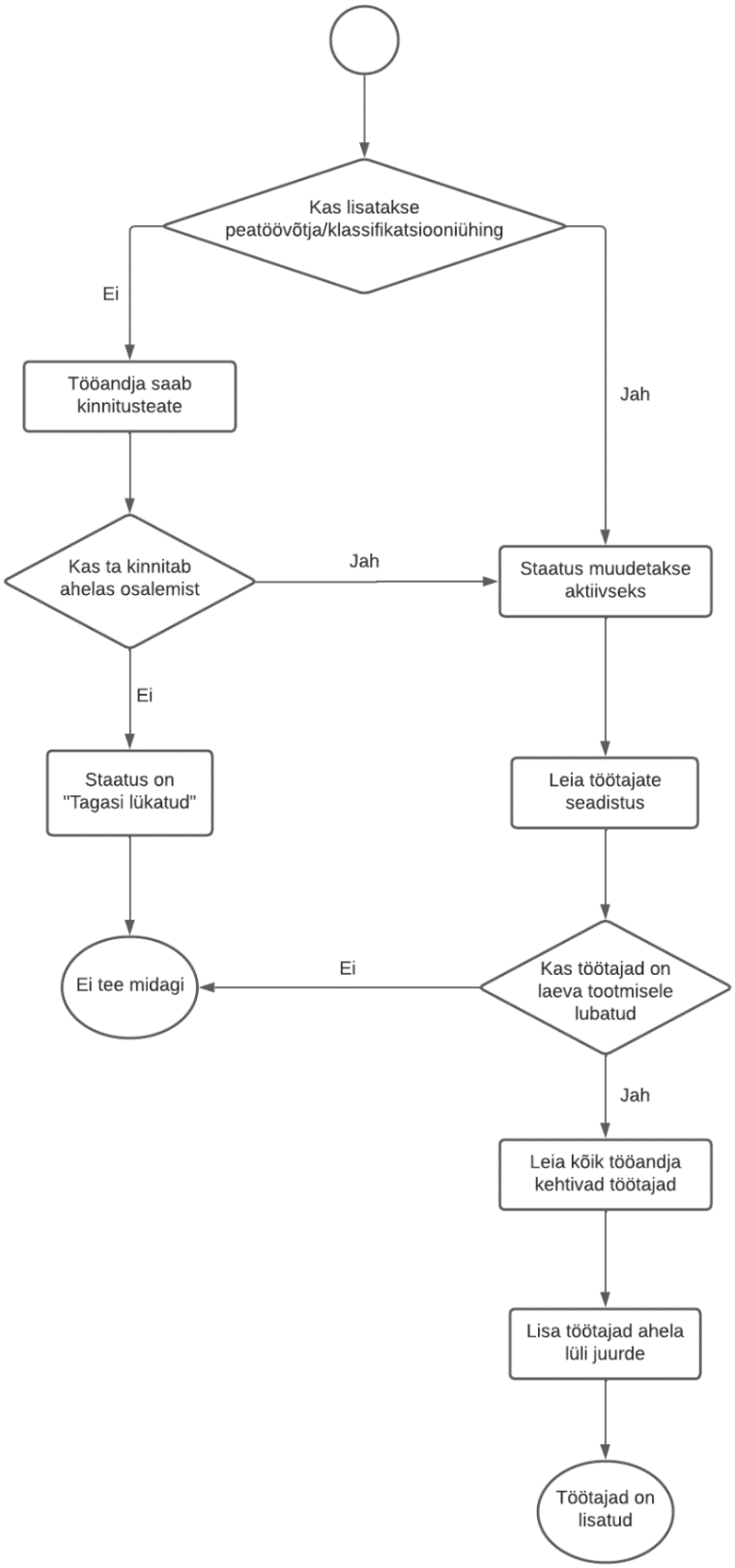


Joonis 6. Töötajate seadistuse leidmise skeem

3.3.2 Tööandja töövõtuahelasse lisamine

Tööandja lisamisel ahelasse antakse kirjele staatus "lisatud". Selline tööandja ei osale veel täielikult LT-s ja sinna töötajaid lisada ei saa. Siiski on sellel reeglil erandeid. Kui töövõtuahela

kirjes pole ülem-tööandjat, tähendab see, et lisati tööandja tasemega 0, mida saab olla ainult PTV, klient ja KÜ. Kui selline kirje luuakse, osaleb tema tööandja juba LT-se protsessis ja neil pole vaja oma osalust kinnitada ning nende staatus on automaatselt aktiivne. Kuna kirje on aktiivseks saanud, siis neile lisatakse töötajad automaatselt kui nende seadistustes on märgitud „jah“. Ühel tööandjal või FIE-l pole piiranguid rollide arvule, mis võib selles etapis viia töötajate kordusteni.

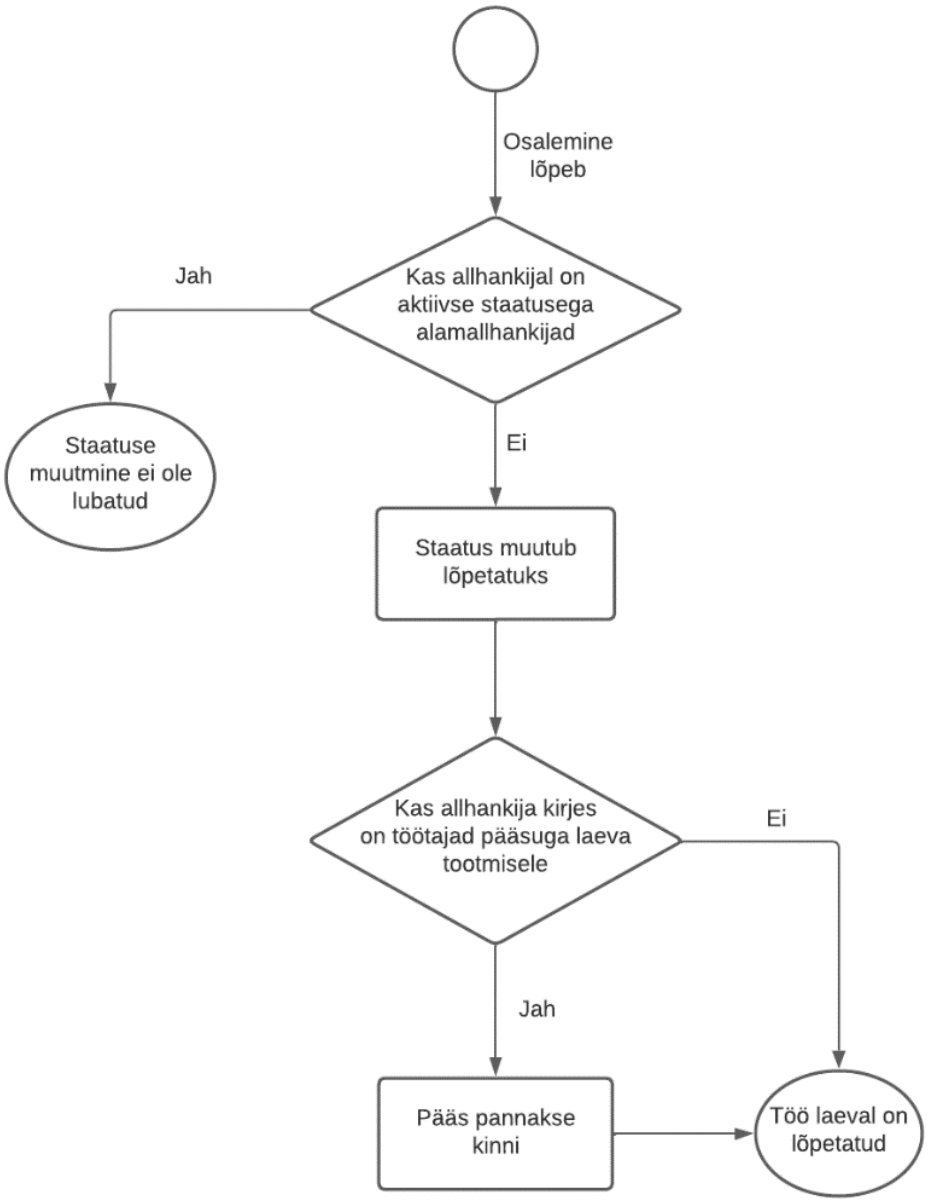


Joonis 7. Tööandja lisamine töövõtuahelasse

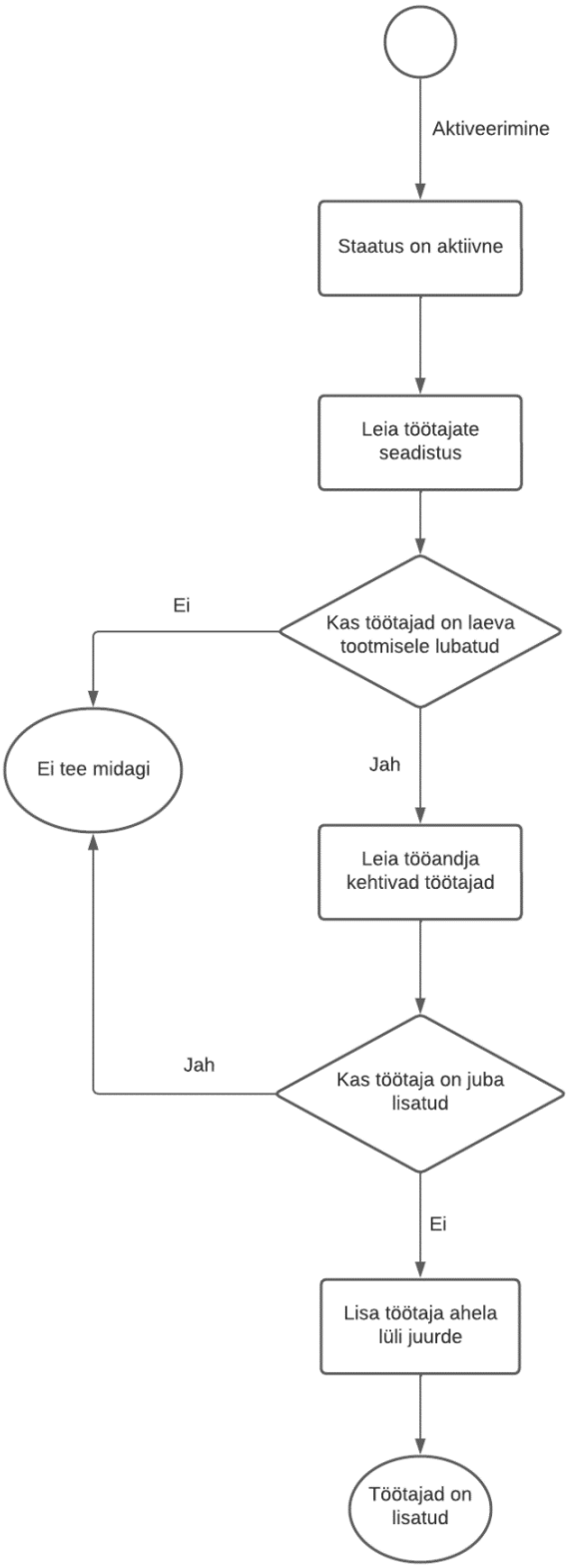
3.3.3 Allhankija staatuse muutmine

Tööandja, kellel pole 0-tasemel staatust kinnitab oma osalemist töövõtuahelas ja kui on seadistus töötajate automaatseks lisamiseks, siis töötajad lisatakse. See võib olla tööandja mitte esimene aktiivne allhankija kirje ja kõik tema aktiivsed töötajad LT-l saavad koopiad.

Aja jooksul lõpeb LT ja tööandjate kirjed tuleb sulgeda. Seda saab teha igal ajal, kui valitud allhankija all pole teisi allhankijaid aktiivsete staatusega. Tööandja või FIE töö lõpetamisel ei tohi nende kõigil töötajatel enam olla pääsu LT-le. Selleks suletakse kõik töötajate kirjed laeva raames, mis on seotud juba suletud allhankijaga. Ühel ja samal tööandjal võib olla üks aktiivne kirje ja teine hiljuti suletud. Kui töötaja lisati varem ainult suletud kirjesse, siis tema pääs LT-le suletakse. Rakenduse eesmärk on hoida valideeritud andmeid ja see ei eelda, et kasutajad avavad seda iga päev ja kontrollivad teavet. Kui töötaja lisati mõlemasse kirjesse, ilmuvad tal kaks täielikult vastandlikku staatust.



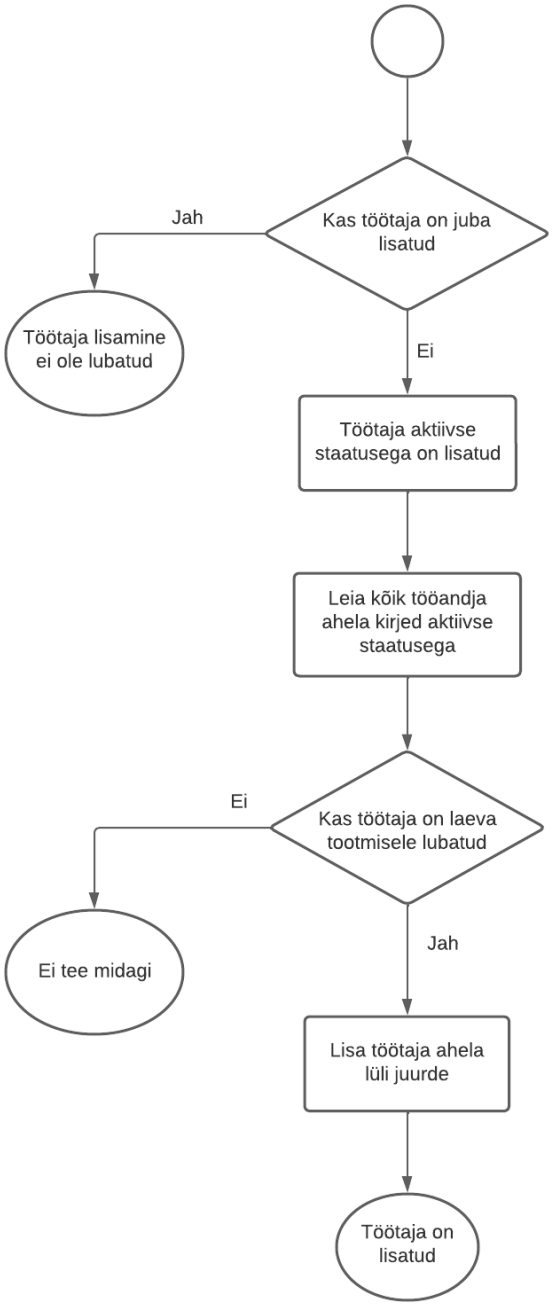
Joonis 8. Allhankija staatuse muutmine. Lõpetamine



Joonis 9. Allhankija staatuse muutmine.
Aktiveerimine

3.3.4 Tööandja üldnimekirja lisamine

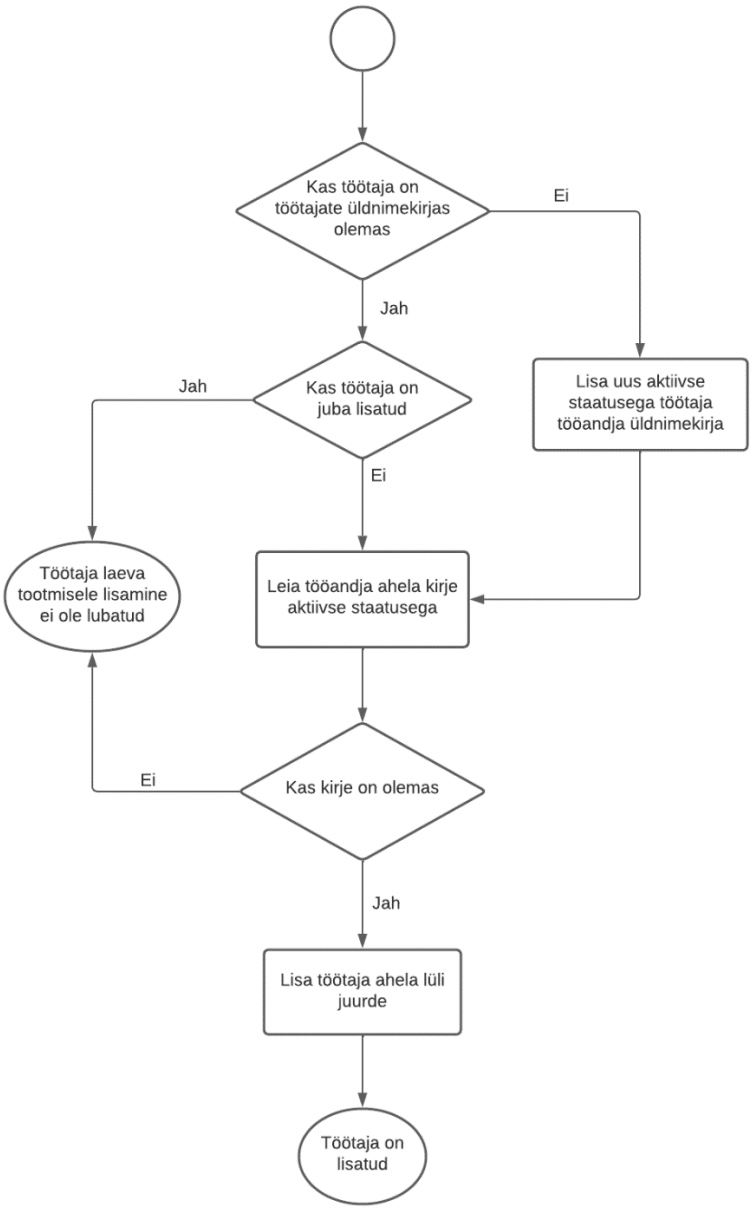
Uue töötaja lisamisel vaatab süsteem kõikide tööandja aktiivseid kirjeid erinevates LT-se protsessides. Töötaja lisatakse automaatselt kõikjale, kuhu see on lubatud. Kui tööandjal on 2 aktiivset kirjet samal laeval, siis töötaja lisatakse sinna kahekordselt.



Joonis 10. Töötaja lisamine tööandja üldnimekirja

3.3.5 Tööandja tootmisele lisamine

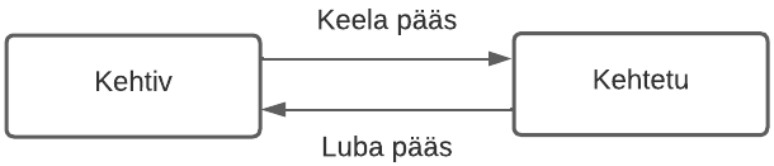
Kui töötajat LT-le lisatakse, siis süsteem ise leiab aktiivse allhankija kirje ja loob selle kirjega seose. Kui ühel tööandjal on kaks kirjet, siis süsteem annab andmebaasi otsimise tasemel veateate. Viga on seotud asjaoluga, et koodis oodatakse ainult ühte kirjet, kuid meetod leiab rohkem. Sellises ebatavalises olukorras süsteem lihtsalt ei tööta ja pole võimalik lisada teisi olemasolevaid töötajaid laevale.



Joonis 11. Töötaja LT-le lisamine

3.3.6 Töötaja pääsu muutmine

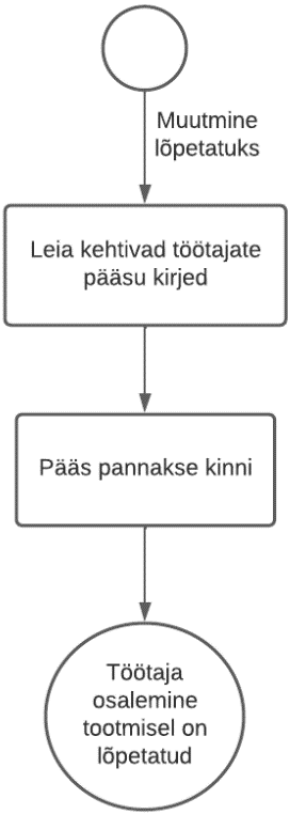
Kui sama töötaja lisatakse LT-le kaks korda, siis pole piiranguid tema pääsu muutmiseks ja üht kirjet on lihtne muuta. Seega ei takista funktsionaalsus töötajal samaaegselt olla kehtiv ja kehtetu pääs.



Joonis 12. Töötaja pääs LT-le

3.3.7 Töötaja staatuse muutmine

Kui tööandja otsustab muuta töötaja aktiivse staatuse mitteaktiivseks, siis süsteem otsib kõik aktiivsed töötaja kirjed kõikidel LT-l ja sulgeb need. Antud funktsionaalsus võtab absoluutselt kõik kirjed järjest, seega töötab õigesti ja seda ei pea parandama.



Joonis 13. Töötaja staatuse muutmine

3.3.8 Töötajate seadistuse muutmine

Kui tööandja otsustab lisada või muuta konkreetse LT-le jaoks töötajate pääsu seadistuse väärtust, siis see muudab kirjete väärtusi tabelis *employment_chain*. Kui tööandjal on mitu kirjet ühe laeva raames, siis tööandja laevade loendis kuvatakse ühe laeva dubleerimised. Vana funktsionaalsus eeldas, et laevaga seotud kirjeid on alati ainult üks ja ei ole mingeid viiteid selle kohta, et laevade dubleerimised on tegelikult seotud erinevate allhankija kirjetega. Kui kasutaja ei märka dubleerimisi ega pane ise kõigile kirjetele sama väärtust *employees_access_allowed*, siis automaatsel töötajate lisamisel laevale lisatakse nad ainult nendesse allhankija kirjetesse, kus väärtus oli tõene. Ülejäänud kirjed jäävad tähelepanuta, mis viib andmete desünkroniseerimiseni.

3.3.9 Töötaja detailandmete vaatamine

Töötaja üksikasjalike andmete lehel on loetelu laevadest, mille LT-l ta on lisatud ja seal näidatakse ka, kas töötajal on pääs või mitte. See leht on mõeldud tööandjale ja kõik pääsud on muudetavad. Kui tööandjal on mitu allhankija kirjet ühe ja sama laeva kohta, siis selles laevade nimekirjas näeb tööandja lihtsalt sama laeva mitu korda, kuid süsteemis on need kirjed täiesti sõltumatud ja sellest lehel midagi ei räägi. Kasutaja saab ühel neist kirjetest pääsu muuta ja andmed hakkavad vastuolu tekitama.

3.3.10 Töötajate nimekiri tootmisel

Töötajate andmete lehel on loetletud kõik töötajad, keda praegune kasutaja saab näha. Ahela ning töötaja nägemise piirangud põhinevad kasutaja paiknemisel ahelas. Igaüks saab näha ainult neid allhankijaid, mis on ahelas tema alla lisatud. Erandiks on ainult PTV, kes näeb absoluutselt kõiki kirjeid laeval, ning klient, kes näeb ka kõiki allhankijaid, kuid ei saa näha ühtegi töötajat.

Kui vaadata kõiki LT-l olevaid töötajaid, ilmub nimekiri, kus on töötaja andmed (ees- ja perekonnanimi), tööandja andmed ning kas töötajal on pääs LT-le või mitte. Nimekirjas ei ole näidatud, millise tööandja kirjega töötaja seotud on. Seega, kui on mitu kirjet, siis on need väliselt kasutajale täiesti identsed. Kui ühega neist muuta pääsu, lubab süsteem seda teha ja pole mingit viidet selle kohta, milline kirje on tõene.

3.3.11 Allhankija deailandmete vaatamine

Eelnevalt mainiti süsteemi, kus ühe tööandja juurdepääs teise töötajate andmetele põhineb hierarhial ahelas. Kui samal tööandjal on kaks kirjet ning allpool teda on veel üks tööandja kahe kirjega, siis ülemine tööandja saab mõlemat näha, kuna ta on mõlemal juhul ahelas kõrgemal positsioonil. Kui alumisel tööandjal on ühe töötajaga mitu kirjet erinevate pääsudega või kui töötaja on ühes kirjes lisatud, aga teises mitte, siis ülemisel tööandjal on kahe erineva tööandja kohta täiesti erinevad andmed töötajate kohta. Alumine tööandja näeb ka oma kahe erineva kirje raames täiesti erinevaid andmeid töötajate kohta. Samal ajal võivad mõlemad ahelas olevad kirjed olla aktiivsed, kuna andmete desünkroniseerimiseks piisab töötaja juurdepääsu muutmisest üldises loendis.

3.4 Probleemide lahendused

Selle ülesande lahendusi võib olla mitu. Mõned neist sobivad paremini ja mõned vähem ja see kõik sõltub mitmest tegurist:

- Ülesanne tuleb lahendada lühikese aja jooksul
- Tuleb minimeerida lahenduse mõju teistele projektis olevatele süsteemidele
- Andmeloendite välimus ja olemus peab jääma samaks
- Iga tabeli rida peaks olema võimalikult informatiivne ja ei tohiks ületada kasutajale kättesaadava teabe piire

Lisaks on mõned reeglid, mis peamiselt käsitlevad lahenduse struktuuri ja koodi kirjutamist. Funktsionaalsus peab vastama projekti üldisele väljanägemisele, tehnoloogiate või nende versioonide muutmine, varem kasutamata raamatukogude importimine või uue projekti struktuuri loomine on keelatud ega ole vajalik. Andmebaasi muudatused tuleb teha uute Liquibase'i skriptide abil, andmebaasi päringud toimuvad repositooriumi abil, seejärel teenused töötlevad neid ja saadavad andmed kliendirakendusele kontrolleri kaudu. Andmetabeli alus on juba olemas kliendirakendusel koos leheküljendusega, seetõttu tuleks selle kasutamiseks lahenduse leidmiseks seda vajadusel muuta.

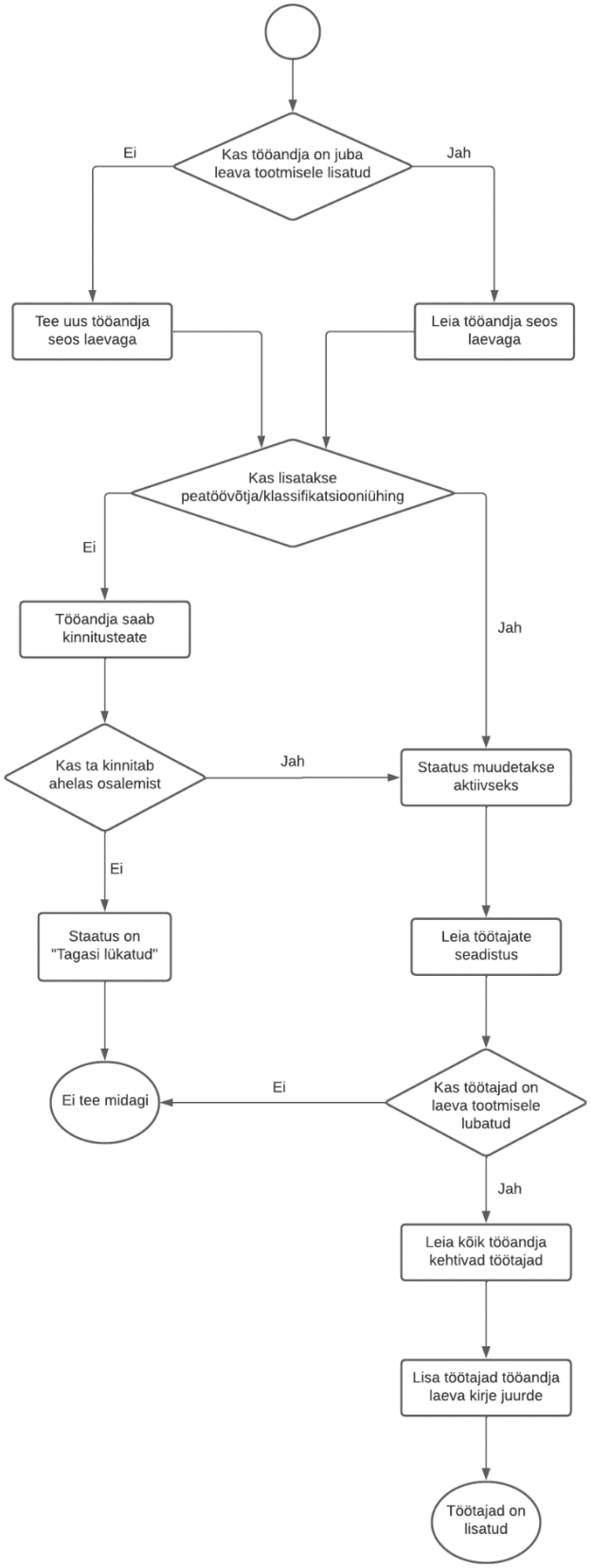
3.4.1 Andmebaaside ümberkirjutamine

Korrektne lahendus andmebaasi vigastele seostele oleks andmebaasi ümberkujundamine. Kui ühendused töötaja ja tööandja vahel ümber korraldada nii, et töötaja võib asuda LT-l allhankija kirje juuresolekul, siis dubleerimist ei teki. Samuti võib tööandja lisamisel laevale tekkida kohe konfliktid *employees_access_allowed* kirjetega. Kui tööandjal on üks põhiline seos laevaga, millest tema seosed ahelas ehitatakse, siis on *employees_access_allowed* väärtus alati ainult üks.

Uue andmebaasi mudeli joonis asub lisas 4.

Uuel skeemil on nähtav, et tööandjal on eraldi ühendus laevaga, kuhu lisatakse töötajad. Selle ühenduse jaoks tekivad ahelas eraldi kirjed. Seega ei mõjuta allhankija kirjed töötajaid ja dubleerimist ei teki.

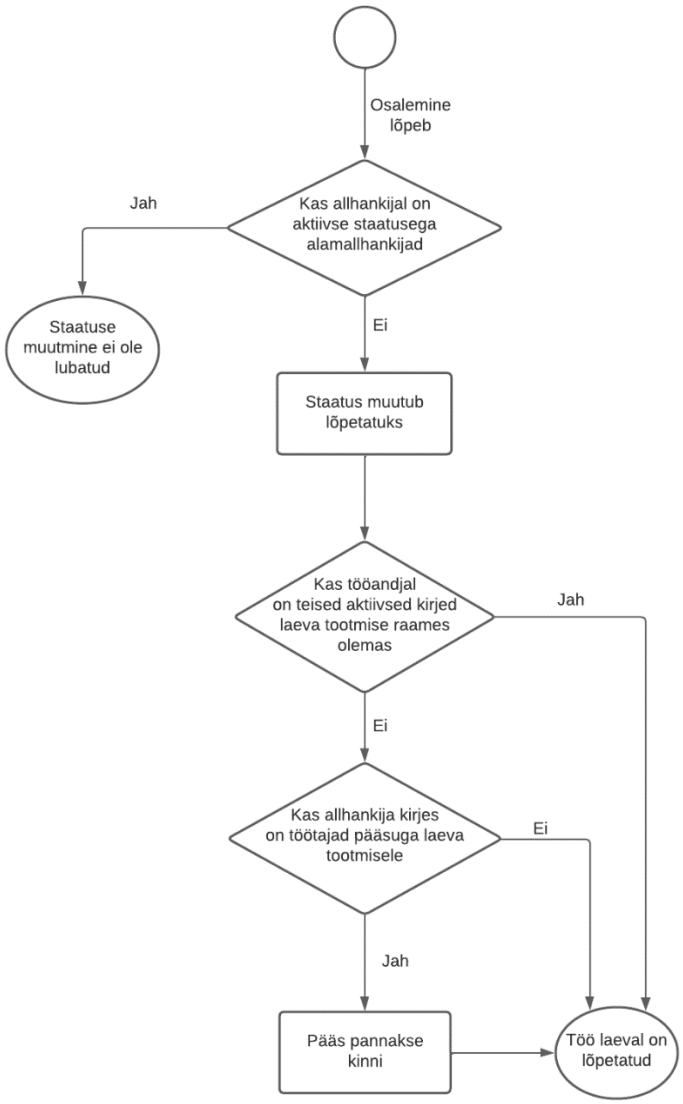
Tööandja lisamisel tuleb kontrollida, kas sellel tööandjal on ühendus valitud laevaga. Kui seda pole, tuleb see esmalt luua. Kui ühendus on juba olemas, võetakse see andmebaasist ja sinna lisatakse tööandja kirje. Ülejäänud funktsionaalsus töötab samamoodi nagu varem.



Joonis 14. Tööandja lisamine töövõtuahelasse. Andmebaaside ümberkirjutamine

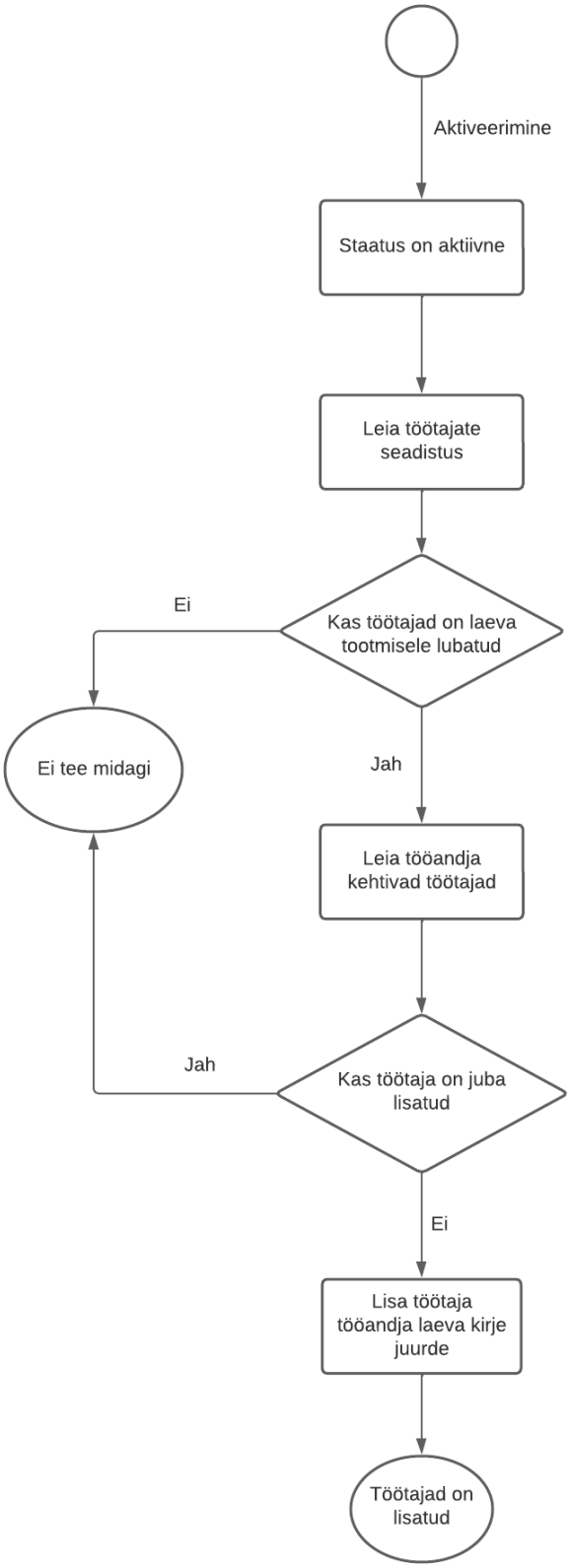
Allhankija staatuse muutmise loogikas pole eriti palju erinevusi. Peamine erinevus seisneb selles, et kui allhankija suletakse, tuleb kontrollida, kas see oli viimane aktiivne kirje. Kui tööandjal pole enam õigusi osaleda LT-s, siis tema töötajatel pole seda õigust samuti. Seega peavad kõik töötajate pääsud sulguma kui viimane allhankija kirje suletakse.

Kui töö jätkub, lisatakse töötajate kirjed tööandja-laeva seosesse, mitte aga ahelasse.



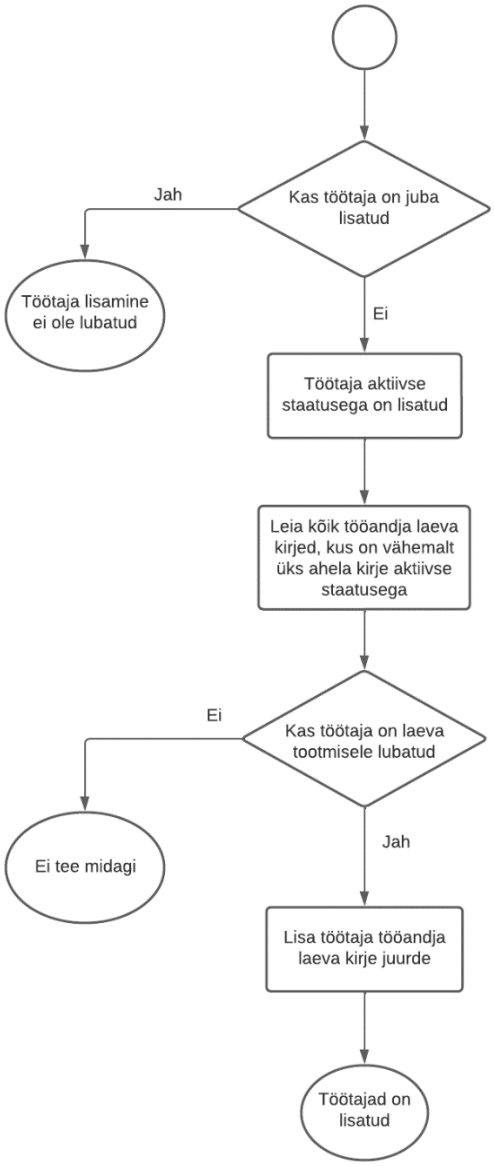
Joonis 15. Allhankija staatuse muutmine. Lõpetamine.

Andmebaaside ümberkirjutamine



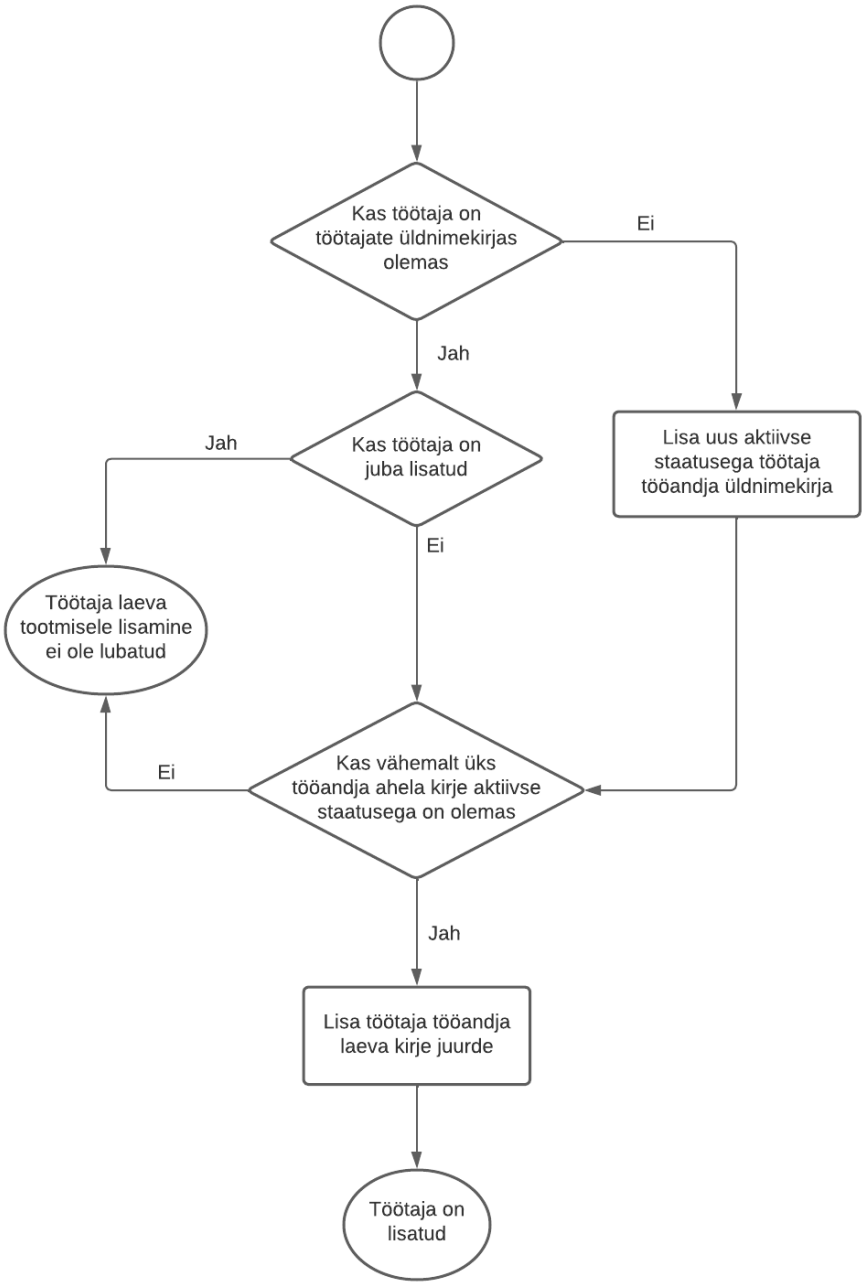
Joonis 16. Allhankija staatuse muutmine.
Aktiveerimine. Andmebaaside ümberkirjutamine

Kui töötaja lisatakse üldisesse töötajate nimekirja, siis otsitakse talle sobivaid laevu töötajate seadete kaudu. Sobivad laevad, kus on ahelas vähemalt üks aktiivne tööandja kirje, filtreeritakse välja ja seejärel lisatakse töötaja tööandja-laeva kirjesse. Ahelate roll viiakse tagaplaanile. Selles süsteemis kasutatakse neid ainult selleks, et näidata, kas tööandjal on hetkel LT-sele pääs. Teostatav tööde liik või "pääsude arv" ei ole tähtis, sest vastusega "jah" piisab ühest kirjest.



Joonis 17. Töötaja lisamine tööandja üldnimekirja. Andmebaaside ümberkirjutamine

Töötaja LT-le lisamise protsess on osaliselt muudetud. Uus töötaja lisatakse üldisesse nimekirja kui teda varem pole olnud, kuid süsteem ei kuva kasutajale viga kui tööandjal on rohkem kui üks kirje ahelas. Kui töötaja lisada LT-le, kus pole ühtegi aktiivset kirjet, saadetakse kasutajale veateade.



Joonis 18. Töötaja LT-le lisamine. Andmebaaside ümberkirjutamine

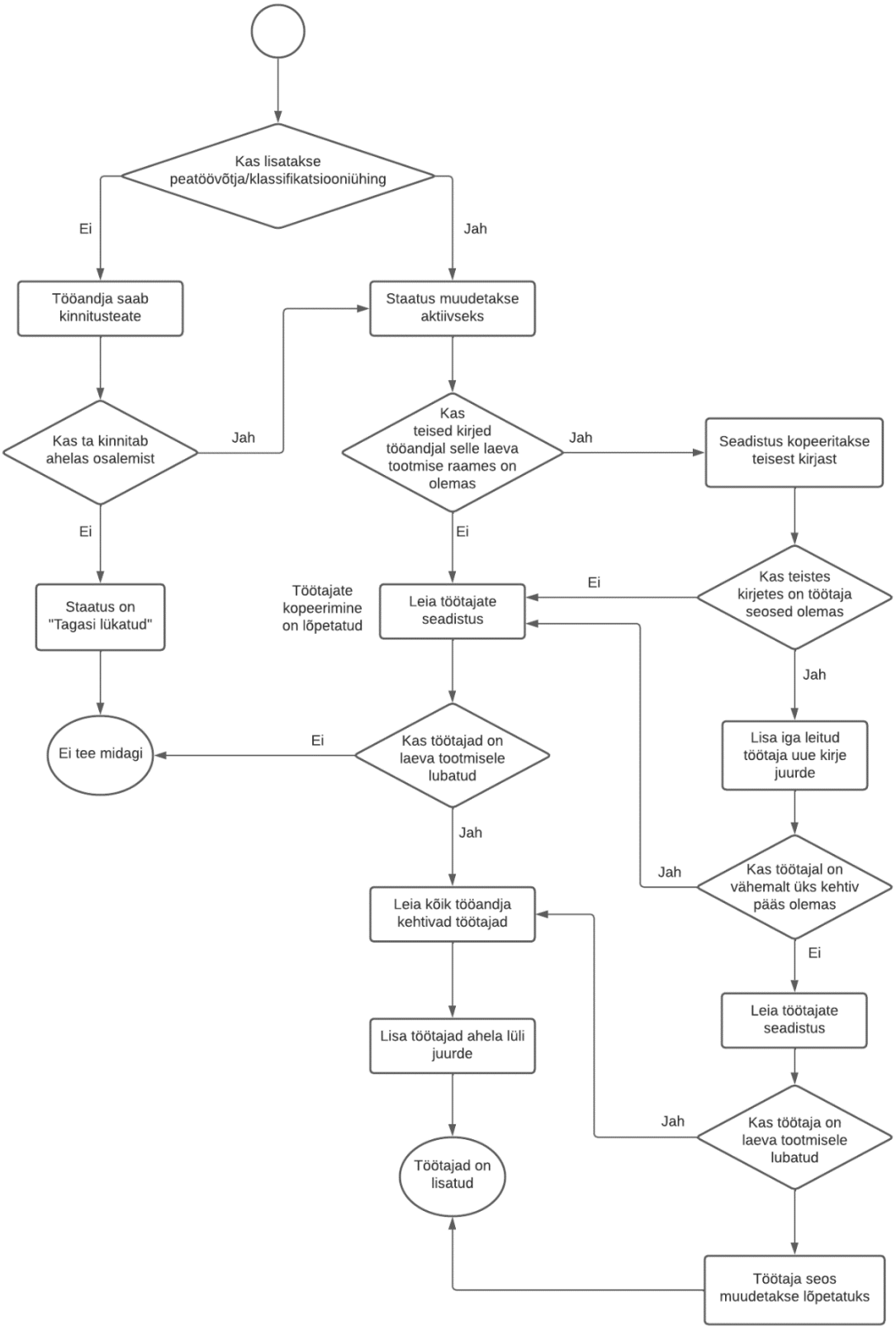
Selle lahenduse peamine eelis on see, et töötaja pääsu muutmise loogikat ei pea muutma. Funktsioonaalsus töötleb alati ainult ühte ainsat kirjet ega tekita topeltkirjeid. Töötaja staatuse muutmine toimib samuti õigesti, kuna kõik tema seosed kõigi varasemate laevadega suletakse täielikult.

Töötajate automaatse lisamise seadetes kaovad topeltandmed, kuna selle kohta salvestatakse teave tööandja-laeva kirjes. Seega õnnestub andmete desünkroniseerimist vältida.

3.4.2 Andmete grupeerimine

Ülejäänud kolme lahenduse ühendab üks asi - need on rakendatavad juhul kui baasi ümberkorraldamine pole võimalik. Põhiline idee seisneb selles, et kõik klassid ja tabelid andmebaasis jäävad samaks nagu need olid algselt. Seega on lahendused keskendunud duplikaatide rühmitamisele. See ei tähenda, et kasutaja peab neid tingimata lehel nägema, kuid need on olemas andmebaasis. Selle süsteemi korrektseks toimimiseks tuleb duplikaadid õigel ajal ja õigete staatustega luua. Iga ahela suhtes on andmed andmebaasis erinevad, kuid küsimusele "Kas töötajal on tootmiseks juurdepääs?" on vastus ainult üks. Kuni töötajal on vähemalt üks kehtiv LT-e pääsu kirje, on vastus "Jah". Kui kõik allhankija kirjed on suletud, siis töötajal pääsu pole.

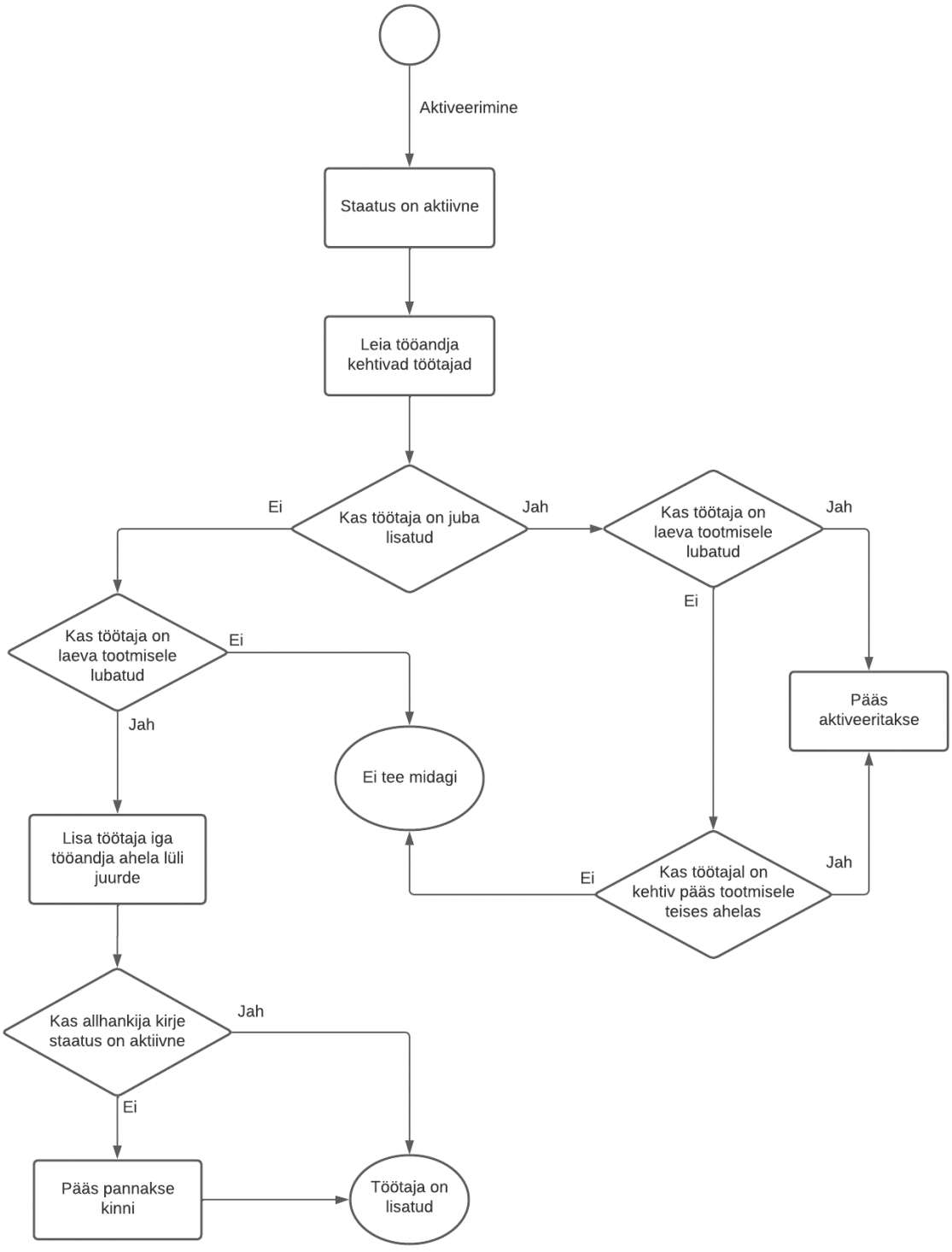
Muudatused loogikas on kõige arusaadavamad tööandja ahelasse lisamisel. Uute aktiivsete töötajate lisamisega peab allhankija kirjetesse kopeerima juba kõik olemasolevad seosed. Seega on kõigi allhankija kirjete vahel pidev sünkroniseerimine töötajate kirjetega. Kui LT-s on juba aktiivsed töötajad ja üks allhankija ning seejärel lisandub teine ahel, siis vanade töötajate sulgemisel ei kaota töötajad juurdepääsu, kuna kirjed on juba uude dubleeritud.



Joonis 19. Tööandja lisamine töövõtuahelasse. Andmete grupeerimine

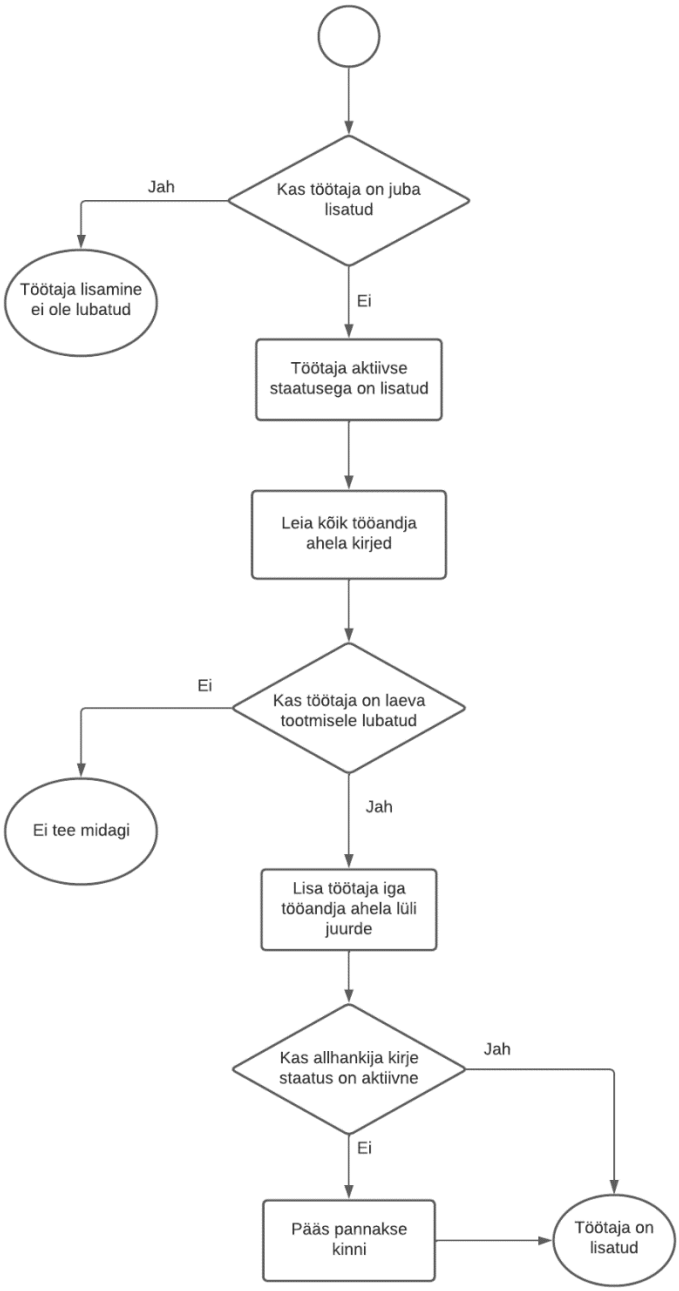
Allhankija kirje sulgemisega jätkub süsteemi töö samamoodi nagu varem. Kõik töötajate pääsud LT-sele suletakse, mis olid varem seotud suletud allhankijaga. Kui on olemas muid aktiivseid alhankijaid, siis töötajatel on ikkagi pääs LT-sele.

Hetkel, mil järjest aktiveeritakse uut allhankija kirjet, peavad varasemalt lisatud töötajate kirjed uuesti aktiveeruma. Kui kõik töötaja muud kirjed on suletud, saab ta uuesti pääsu LT-le ainult siis kui seadistus võimaldab töötajaid automaatselt lisada. Kui seda pole, siis jääb töötaja endiselt suletuks. See on tehtud selleks, et mitte kustutada kasutaja käsitsi suletud pääsu LT-le. Automaatne aktiivsete töötajate lisamine töötab samamoodi nagu varem. Ainult mitteaktiivsed alhankijate kirjed suletakse kohe. Töötajal ei saa olla pääsu kui tema tööandja kõik kirjed on suletud. Kuna allhankijad suletakse ükshaaval, siis suletakse ka vastavate töötajate kirjed eraldi.



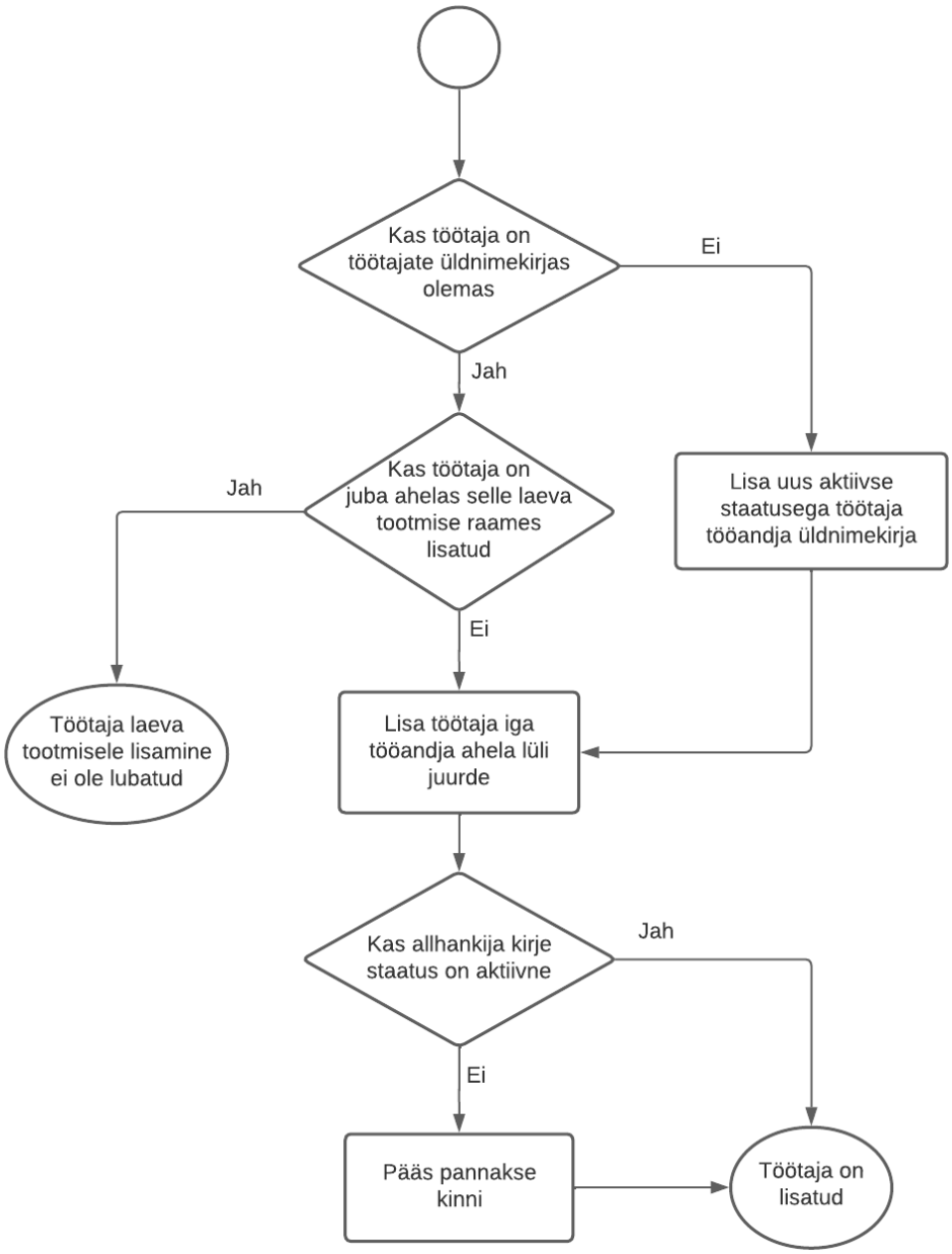
Joonis 20. Allhankija staatuse muutmine. Aktiveerimine. Andmete grupeerimine

Sarnaselt eelmistega töötab ka järgmine meetod: kui töötaja lisatakse üldisesse loendisse, siis tööandja töötaja ilmub kõigis allhankijate kirjetes, kus automaatne lisamine on lubatud. Kui allhankija kirje pole sel ajal aktiivne, siis töötaja kirje suletakse automaatselt.



Joonis 21. Töötaja lisamine tööandja üldnimekirja. Andmete grupeerimine

Töötaja LT-le eraldi lisamisel andmeid lisatakse iga tööandja allhankija kirjesse.



Joonis 22. Töötaja LT-le lisamine. Andmete grupeerimine

Töötaja pääsu muutmisel LT-le muutuvad andmed ainult aktiivsetes allhankijate kirjetes. See võimaldab vältida pääsu töötajatele, kelle allhankijate kirjed on suletud. Kui ühel ja samal tööandjal on üks avatud ja üks suletud ahel, muudab see töötajate pääsu ainult nendele kirjetele,

mis võimaldavad andmete muutmist. Sel juhul näeb töötaja mõlemas andmete kirjes välja ühesugune, kuna tema pääs arvutatakse, mitte ei võeta lihtsalt allhankijate kirjest. Kui töötaja staatus muutub mitteaktiivseks, sulgeb süsteem kõik tema aktiivsed seotud kirjed LT-l.

Automaatse töötajate lisamise seadistuse muutmine LT-l nõuab täiendavat tööd. Kuna see teave on iga allhankija jaoks eraldi salvestatud, tuleb seda väärtust muuta kõigis LT-se protsessi raames olemasolevates kirjetes. Varem mainitud funktsionaalsuse abil uue tööandja kirje lisamisel ahelasse võetakse see väärtus teistest olemasolevatest kirjetest kui need on olemas. Kui väärtust ei ole määratud või muid kirjeid pole olemas, jääb väli tühjaks ja kasutaja saab selle hiljem käsitsi sisestada. Kui sel ajal on juba mitu kirjet, saavad need kõik sama väärtuse. Seega on vastus küsimusele "Kas töötajale on lubatud lisada LT-le?" ühesugune olenemata sellest, millise allhankija kirje jaoks ta küsib.

3.4.3 Serverirakendusel grupeerimine koodis

Andmete grupeerimine serverisakensuses võimaldab mitte ainult hoida kliendirakenduse osa muutmata ja hoida komponente puutumata. Selline meetod võimaldab kasutada Java ja Springi võimalusi, kus andmete töötlemiseks kulub vähe aega ning koodi ei koormata üle.

Selle lahenduse põhiolemus seisneb selles, et kõik duplikaatide töötlemise kohustused kantakse üle EmploymentChainEmployeService teenusele. Päringu struktuur on järgmine:

1. Klinedirakendus genereerib lehe komponendiga, kus peaks olema nimekiri kõigist töötajatest LT-l, keda kasutaja saab vaadata ja muuta.

2. Kasutajaliides kuvab andmete laadimise ikooni, saadab päringu serverirakendusele.

3. Päring koostatakse konkreetse laeva suhtes ja seda töötleb EmploymentChainEmployeeController

4. Kontroller teisendab filtrid DTO-ks (*Data Transfer Object* - andmeedastusobjekt) ja kutsub välja meetodi EmploymentChainEmployeeService, mis leiab töötajad.

5. EmploymentChainEmployeeService kontrollib kasutaja pääsu ja päringu andmete kehtivust. Kui kasutaja pole valitud LT-l tööandja, siis tal pole õigust töötajate nimekirja

küsida. Sellised kontrollid on igas meetodis ning teenuses, mida kontrollerid kasutavad. Seega katab turvareeglite võrgustik rakenduse kõik meetodid ja viga kuvatakse enne kui arvutuse või kehtivuse kontrolle alustatakse.

6. Meetod teenuses küsib andmeid repositooriumilt, seejärel filtreerib kehtetud andmed, teisendab vastuse DTO-ks ja tagastab kontrollerile.

7. Kontroller saab loendit teenusest, teisendab tulemuse mudeliks ja vastus saadetakse tagasi kliendirakendusele.

8. Kliendirakendus saab andmed ja kuvab täielikult kõik, mis serverirakendusest tuli. Kliendirakendus ei tee pääsu kontrolli, sest kõik on juba õiges vormingus serverirakenduste vastusega saabunud. Kui teave on piiratud ainult komponentidega kliendirakenduses, siis saaks iga kasutaja avada arendaja tööriistad brauseris ja vaadata rohkem kui tal selleks õigus oli, mis aga pole lubatud ega turvaline.

Kõige keerulisem selle lahenduse juures on töötajate filtreerimine, keda kasutaja võib näha, sest töötajate hulgas on ka kirjeid, kes temaga seotud ei ole. Isegi ühel laeval näeb tavaline allhankija ainult oma töötajaid ja neid, kes on tema all tööandjate ahelas. Ainuke erand on PTV, kes näeb kõiki ahelaid ja kõiki töötajaid, kuid keegi teine peale teda ei saa näha informatsiooni teistest ahelatest ega oma ahelast kõrgemal tasemel.

Kuna kõik LT töötajad on seotud tööandjatega ja pääsu kontrollimine algab just ahelas, siis töötajate filtreerimisel tasub neid grupeerida tööandjate suhtes. Kui kasutajal on pääs konkreetse tööandja andmetele, siis tal on pääs kõikidele tema töötajatele. Seega saab päringu "kõikidele LT töötajatele" muuta päringuks "kõikidele LT töötajatele, kelle tööandja on antud loendis" ning loendis on tööandjad, kellel on pääs.

See tähendab, et esialgu tuleb teada saada olemasolevate tööandjate loend, seejärel välja tuua kõik nende töötajad, neid grupeerida ja seejärel kustutada duplikaatide kirjed.

Filtreerimine keerukama loogikaga võib toimuda otse andmebaasi repositooriumi meetodis, mis suurendab kogu süsteemi töö kiirust. Kõike ei saa sellisel viisil filtreerida, aga näiteks töötaja, tööandja nime või isikukoodi saab.

Sellisel viisil saab toimida meetod, mis võtab kõik kasutaja jaoks saadaolevad töötajad LT-l.

1. Vajalik on koostada nimekiri kõigist allhankijatest LT-l. Filtreerige loend ja jätke alles ainult need kirjed, millele kasutajal on pääs. Võtke välja tööandjate id-d ahelatest.

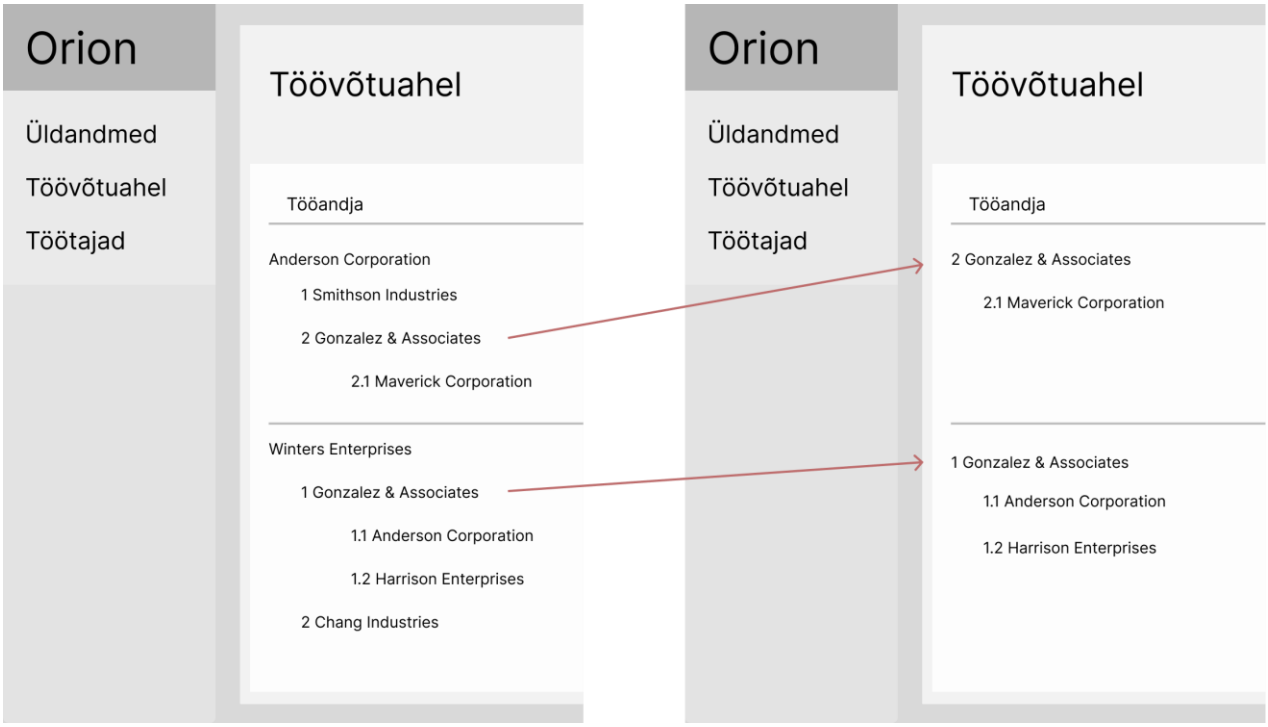
2. Vajalik on teha päring andmebaasist valmis nimekirja kättesaadavate tööandjate ja töötajate filtreerimise parameetritega. Vastus grupeeritakse töötaja järgi.

3. Iga töötaja jaoks on nimekiri tema tööandja töösuhete kirjetest. Nüüd tuleb sellest kirjete rühmast teada saada, kas töötajal on pääs LT-le. Tsükli ajal on vajalik teisendada iga töötaja kirje DTO-ks ning vastus töötaja pääsu kohta LT-le salvestatakse DTO-sse *booleanina*.

See lahendus on võrdlemisi mahukas. Algselt on vaja tööandjate nimekirja, seejärel on vaja teha muutujate filtreerimiseks päring repositooriumis. Seejärel tuleb saadud vastus grupeerida ja filtreerida veel kord.

3.4.4 Kliendirakendusel grupeerimine

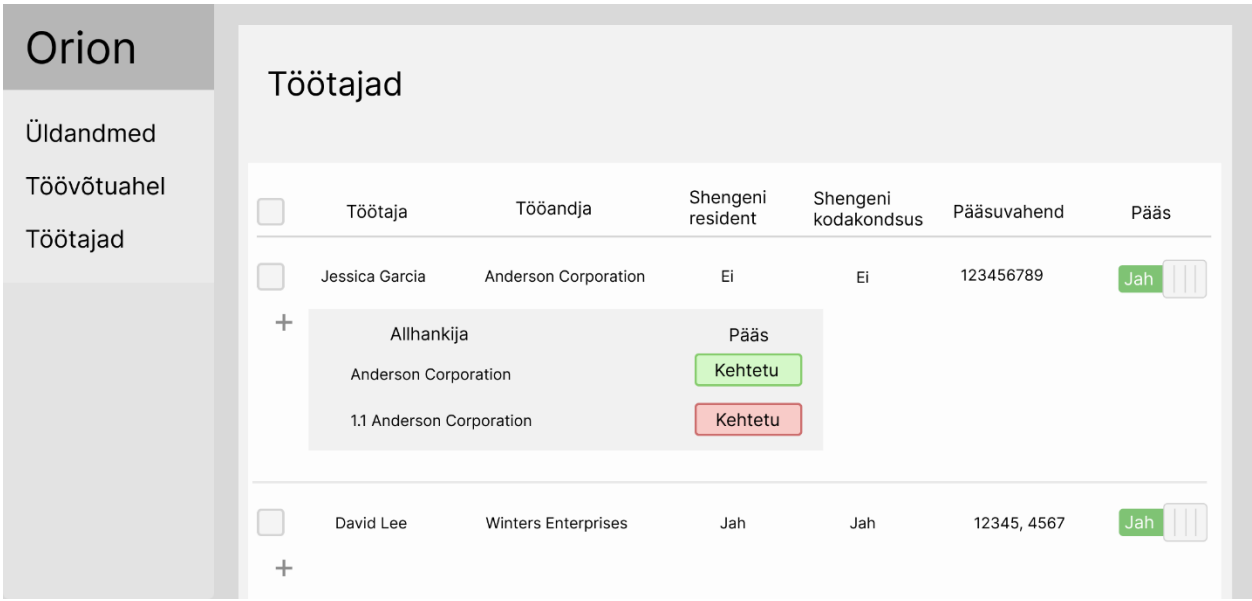
Järgnev andmeanalüüsi töötlus sarnaneb eelmise lahendusega võimaldades kergelt muuta, kuidas andmed kasutajatele edastatakse. Kuna töötajate teavet saavad vaadata ainult tööandjad, kes on ahelasse lisatud, siis kasutajal on arusaam, kus ta hierarhias asub ning ta teab, kas teda on sinna lisatud kaks või isegi kolm korda. Kui tööandja pole ahela alguses, siis näeb ta seda enda juurest allapoole ning näeb ka lihtsustatud kirjet selle kohta, kes on tema üleval.



Joonis 23. Töövõtuahela PTV (Anderson) vaade. Töövõtuahela allhankija (Gonzalez) vaade

Selle lahenduse peamine idee seisneb selles, et kasutaja näeks oma töötaja kõiki kirjeid kõikides allhankija kirjetes, kuid need kirjed on grupeeritud ja näitavad töötaja "lõplikku" pääsu LT-le.

Iga töötaja nimekirjas on täiendav plokk, kus on loetletud tema seosed allhankijatega. Seal on näidatud kirje staatus, töötaja staatus ning kas selle konkreetse kirje kaudu on pääs LT-le. Töötaja kirje ise üldises nimekirjas näeb täpselt välja nagu vanas funktsionaalsuses ning kui klikkida nupul "+", avaneb paneel täiendava teabega. See on dekoratiivne plokk, kuid võimaldab tööandjal kokku võtta kogu töötaja kohta käiva informatsiooni LT-sel. Nii on kogu töötajaga seotud teave grupeeritud ühes kohas ja võimalikult detailselt. Andmete töötlemiseks tuleb avada teisi tabeleid, kuid välja toodud töötajate nimekirja kujunduse variant on efektiivne.



Joonis 24. Lisakomponent pääsuga iga allhankija kohta

See lahendus vabastab süsteemi eelmise lahenduse meetodist töötajate päringutele LT-l viimasel sammul. Selle sammu eesmärk oli arvutada vastus küsimusele "Milline on töötaja kogu pääs LT-le?". Nüüd pole seda sammu vaja ja andmed tuleb iga töötaja jaoks grupeerida ja teisendada üksikasjalikumaks DTO-ks, kuna nüüd kuvatakse ka ahelate staatuste informatsioon. LT-le juurdepääsu üldine staatus arvutatakse kliendirakendusel, kuna andmed saabuvad serverist iga töötaja jaoks grupeerituna ning õige staatuse kuvamiseks saab luua komponent.

3.4.5 VIEW tabeli grupeerimine

Probleemi lahendamise hetkel täiendav tingimus ei olnud lisatud lõplikku nimekirja, kuid lahenduse valimisel tuleb arvestada süsteemi võimaliku arenguga. See täiendav tingimus on enamiku rakenduse olemasolevate nimekirjade pagineerimine ja sorteerimine. Filtreerimine on rakenduses juba pikka aega kasutusel ja LT töötajate funktsioonid töötavad ka sellega. Filtreerimine rakendatakse selle lahenduse puhul juba päringu etapis andmebaasi, st sobimatud väljad kõrvaldatakse võimalikult enneagselt. Samuti rakendatakse seda ka tööandjate nimekirjas, kellele kasutajal on pääs, st päringus andmebaasi töötajad filtreeritakse nende tööandjate järgi, keda kasutaja võib vaadata. Paginatsiooni saab rakendada juba teenuses ja pärast päringut andmebaasi, kuid see tähendab, et süsteem võtab kõik saadaolevad ja sobivad kirjed, teisendab

need ja arvutab kõigi töötajate pääsud LT-dele. Seejärel võtab ainult esimesed 10 töötajat nimekirjas ja annab need edasi kliendirakendusele. Sellise süsteemi korral tähendab iga päring palju rohkem tööd kui tegelikult vaja on. Eelmised kaks lahendust ei ole valmis sorteerimise ja paginatsiooni lisamiseks kasutades liigselt ressursse kui andmebaasi ümberkirjutamise võimalus.

Selle lahenduse variandi loomisel valmistatakse süsteem ette tabelite täiustamiseks. Põhiolemus seisneb selles, et kõik arvutused, mis ei puuduta kasutaja pääsu, viiakse üle andmebaasi. Sel viisil on vaja teha ainult üks päring andmebaasi, teisendada tulemused DTO-le ja saata need kontrollerile.

Selle idee elluviimiseks sobivad VIEW (*Virtual table whose contents are defined by a query* - virtuaalne tabel, mille sisu on määratletud päringuga). VIEW saab kasutada kui tabelit andmebaasis ja sealt andmeid pärida. Samal ajal ei sisalda ta ise andmeid, vaid küsib neid teistest tabelitest päringu ajal. Seega võib virtuaaltabel koguda kogu teavet töötajate kohta ilma duplikaatideta ja juba sellest valmis tabelist saab võtta õigesti filtreeritud, sorteeritud ja pagineeritud andmeid. Samuti virtuaaltabel võimaldab töödelda ja pärida ainult seda piiratud andmegruppi, mis on vajalik konkreetse loendi jaoks rakenduses. [23] [24] [25]

Laevatehase töötajate andmete loendites kasutatakse algselt 5 peamisest tabelist pärit andmeid:

- employee – töötaja nimi, perekonnanimi ja muud isiklikud andmed
- employer – tööandja nimi, tema isikukood
- employer_employee – töötaja staatus tööandja juures, tema pääsuvahendid
- ship – laeva nimi, IMO number ja registreerimisriik
- employment_chain_employee – kas töötajal on juurdepääs LT-le

Virtuaaltabelis saab grupeerida esimesed 4 tabelit ja arvutada pääsu dünaamiliselt. Seega ei kasutata *employment_chain* ja *employment_chain_employee* tabelit otseselt, kuid VIEW küsib neid tabelitelt seoste loomiseks teiste tabelitega ja pääsu määratlemiseks.

3.4.6 Kokkuvõte

Kõik võimalikud lahendused probleemile on läbi vaadatud. Kokkuvõtteks on kõik lahendused grupeeritud ühte tabelisse, nii et saab kindlaks valida kõige sobivama lahenduse. Iga lahenduse kriteeriumi vastas on märge "+" (sobib), "+-" (osaliselt sobib) või "-" (ei sobi).

Tabel 1. Lahenduste eelised ja puudused

	Andmebaaside ümberkirjutamine	Grupeerimine servarirakendusel koodis	Grupeerimine kliendirakendusel ja disaini lahendus	Grupeerimine SQL lausega
Väike ajakulu rakendamisel	-	+	+	-
Koodi kvaliteet	+	-	-	+-
Funktsionaalsuse töö kiirus	+	-	-	+
Funktsionaalsuse mugavus edasiste täiustuste jaoks	+	-	-	+-
Lahendus ei mõjuta teisi funktsioonaalsuseid	-	+	+	+
Iga rida nimekirjas on maksimaalselt informatiivne	+	-	+	-

Kõige õigemaks lahenduseks võib nimetada andmebaasi ümberkirjutamist. Selliste probleemide lahendamisel ei saa kindel olla, et see funktsioon pärast ei kasuta muid süsteeme. Virtuaalne tabel

võimaldab vajalikud andmed võtta ilma dubleerimiseta, kuid see on piiratud valitud väljadega ja teise välja lihtsalt lisamine pole sellesse tabelisse võimalik, nagu tavalisse tabelisse. Kuna projektis kasutatakse Liquibase'i, tuleb igat muudatust salvestada uude skripti. Virtuaalse tabeli puhul see kustutatakse ja luuakse iga kord uuesti.

Kuna kogu tööandja lisamise funktsionaalsus LT-le oli juba rakendatud ja saadetud klientidele testimiseks, tähendab andmebaasi muutmise variant tagasiminekut mitme sammu võrra. Kõik tööandjate kirjed testkeskkonnas laevalt tuleb kustutada ja uuesti täita, kõik testid tuleb uuesti läbi viia ja uued testiraportid koostada. Kui midagi uuesti kontrollimata jätta, võib hiljem avastada vea, mille avastavad ainult juba reaalsed kasutajad. Testikeskkonna andmete kustutamine pole sama kriitiline kui olemasoleva tootmise puhul. Siiski muutis andmete kustutamise ja uuesti testimise vajadus selle probleemi lahenduse vähem atraktiivseks. Lõpuks otsustati sellest lahendusest loobuda just teiste funktsioonide mõjutamise tõttu.

Tulevaste tabelite parenduste ja pagineerimise võimaluse tõttu võib andmete grupeerimine pärast nende väljavõtmist andmebaasist oluliselt mõjutada rakenduse jõudlust. Kuna selle probleemi lahenduse valik peaks olema suunatud jõudlusele, võib töötaja igale ahelale lisainformatsiooni mitte kuvada.

Lõpuks rakendati VIEW lahendus. See on midagi keskmist kõigi valitud lahenduste vahel ja on sobiv antud projekti kokkupressitud ajakava ja äriprotsessi tingimustes.

4 Valitud lahendus

Valitud lahenduse rakendamiseks on vaja teha muudatusi nii olemasolevas koodis kui ka lisada uusi meetodeid ja klasse. Selles osas kirjeldatakse valitud lahenduse rakendamist nii serverirakendusel kui ka kliendirakendusel. Kuna kõik nimekirjad projektis on koostatud ühe skeemi järgi, siis selles osas käsitletakse ainult ühe LT-le töötajate nimekirja lehte.

4.1 Kliendirakenduse osa

Vaatamata ulatuslikele muudatustele funktsionaalsuse loogikas jääb osa tööst kliendirakendusel praktiliselt puutumatuks. Nimekirjas on vaja värskendada andmevälju ja lisada uusi komponente uute funktsioonide realiseerimiseks. Üks sellistest funktsioonidest on suutlikkus valida korraga mitu töötajat ja muuta neile pääsu korraga.

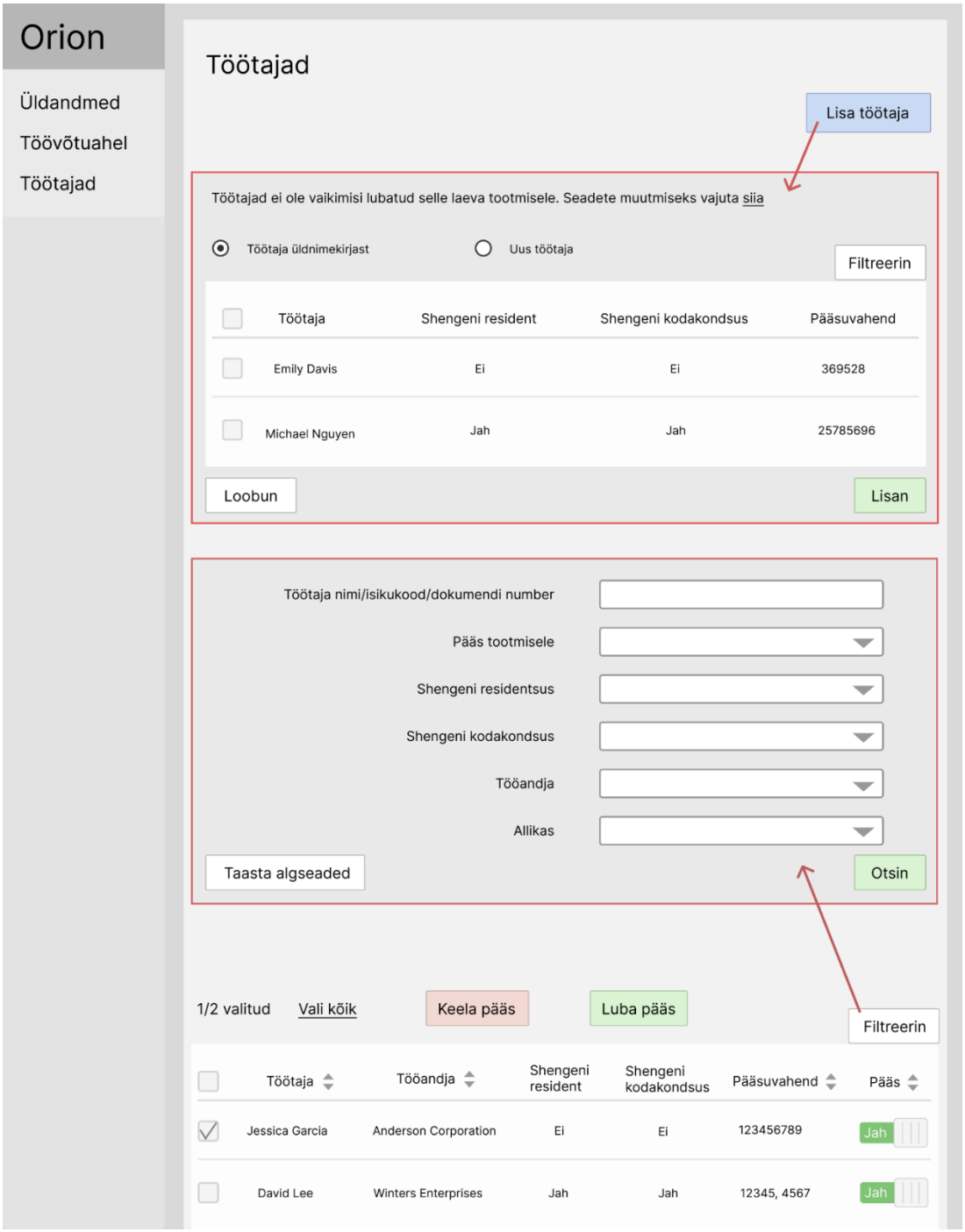
4.1.1 Veebilehte disaini demo

Andmeanalüüsi tulemuste põhjal peaks töötajate lehel iga töötaja kõrval olema märkeruut. Kui mõni töötajatest on valitud, ilmuvad nupud "luba pääs" ja "keela pääs". Samuti peaks lehel olema nupp "vali kõik" ja selle vajutamisel ilmuvad samamoodi nupud pääsu valimiseks. Lehel on filter, mida tuleb värskendada ja asendada mõned andmeväljad. Kui filtril on täidetud mistahes väljad ja kasutaja soovib seejärel muuta kõigi töötajate staatust, siis peaks muutuma ainult nende staatus, kes vastavad filtri kriteeriumidele. Kui kõik töötajad on valitud nupu "vali kõik" abil ja seejärel tühistatakse mõne töötaja märkeruut, muutub staatust kõigil, välja arvatud neil, kellel oli märkeruut eemaldatud.

Iga töötaja juures on kirjas, kas ta on Schengeni riigi kodanik või mitte. Kõik väljad peale riikide peavad toetama sorteerimist. Pääsuvahendite väljad peavad sisaldama nii töötaja kui ka tööandja-töötaja dokumente. Neid näevad kõik, kes saavad seda töötajat nimekirjas näha. [26]

Eelmises funktsionaalsuse versioonis oli igal töötajal *toggle,* mille vahetamisel muutus pääs ainult ühel konkreetsel töötajal.

Peamise töötajate nimekirja kohal on nupp, mis võimaldab lisada uue töötaja LT-le. Töötaja saab lisada valides juba olemasolevate tööandjate töötajate nimekirjast või luues täiesti uue töötaja nullist. Seal näidatakse seadistuse väärtust ja olemasolevatel töötajate nimekirjal on filter.



Joonis 25. Töötajate nimekiri LT-l. Lehte disaini demo

4.1.2 Komponendid

Kõik komponendid asuvad ühes peablokis (peamine komponent). Seal asuvad nupud "uue töötaja lisamine", "kõigi valimine", "pääsu lubamine" ja "pääsu keelamine". Seal initsialiseeritakse valitud töötajate tühi loend ja väli, mis näitab, kas kõik töötajad on hetkel valitud. Samuti küsitakse seal töötajatele valitud LT pääsu seadistust. See komponent on koht, kus asub meetod töötajate pääsu LT-le muutmiseks, samuti meetodid töötajate lisamiseks LT-le.

Peamises komponendis kasutatakse kahte järgmist plokki: plokk töötajate lisamiseks LT-le ja plokk, kus kuvatakse juba lisatud töötajate loend. Mõlemad komponendid sisaldavad endas põhilisi plokke koos filtriga.

Töötajate lisamisel edastatakse teavet selle kohta, millise laevaga hetkel töötatakse, samuti seadistus LT-le. Komponendil on meetod, mis küsib kõiki valitud tööandja töötajaid, kes varem ei ole seotud valitud laevaga. Samuti kasutatakse seal meetodit, mis küsib tööandja kõiki saadaolevaid pääsuvahendeid filtreerimiseks.

Varasemalt lisatud töötajate komponendis edastatakse valitud töötajate nimekirjad eelmisest komponendist, samuti sisaldab see meetodit, mis küsib töötajaid LT-l ja salvestab tulemused. See komponent on mõeldud eraldi teabe päringute jaoks filtrist ning eraldatud loogiliselt, et eraldada filter otsingu tulemustest.

Kahel filtri komponendil on olemas ka kaks töötajate nimekirja komponenti. Seal on tabeli päised, mis seadistavad sorteerimist ja pärivad esimesest komponendist filtreerimise meetodit kui selle peal vajutatakse. Samuti kuvatakse iga töötaja vastas märkeruut ja väärtus salvestatakse esimese komponendi töötajate nimekirja. Filtri tulemusi kasutatakse just siin.

4.1.3 Teenused ja päringud

Kõik kasutatud komponendid kokkuvõtes kasutavad selliseid meetodeid:

1. Töötajate lisamine LT-le nullist POST
2. Töötajate lisamine LT-le valitud nimekirjast POST
3. Täätajate pääsu muutmine (filtreerimise ja töötajate nimekirjaga) PUT

4. Töötajate automaatse pääsu seadistuse laadimine GET
5. Aktiivsete töötajate loendi laadimine, kes on lisamiseks saadaval GET
6. Pääsuvahendite laadimine töötajate filtreerimiseks (lisamine loendist, valmis loendi vaatamine) GET
7. Kõigi juba lisatud töötajate laadimine GET
8. Tööandjate filtreerimiseks saadaval olevate tööandjate laadimine GET

4.2 Serverirakenduse osa

Funktsionaalsuse õige toimimise tagamiseks tuleb serverirakendusel rakendada kõiki meetodeid, mida vajab kliendirakendus. Varasemalt olemasolevad meetodid vajavad töötajate duplikaatide õiget kuvamist. Kõige märkimisväärsemad muudatused on meetodites, mis tagastavad töötaja-laeva andmete kombinatsiooni. Edasi käsitletakse LT-le lisatud töötajate nimekirja päringu tegemist.

4.2.1 Andmebaas

Kõik andmebaasi muudatused tehakse Liquibase'i abil. Töötajate kuvamiseks ilma duplikaatideta tuleb luua uus SQL *changeset* ja lisada see üldisesse *changelogi*. Uue faili jaoks peab olema seadistatud relativeToChangelogFile=true, nii et järgmise andmebaasi värskendamise ajal (gradle -> Liquibase -> dbUpdate) leitakse uus fail, mis varem oli käivitamata ja see käivitatakse. Koos sellega lisatakse see kirje andmebaasi ja järgmisel korral andmebaasi värskendamine ignoreerib juba käivitatud *scripti*. VIEW loomine on kirjutatud eraldi SQL faili (10-create-ship-employee-view.sql), mis on kirjeldatud peatükis Lisa 2. [27] [28] [29]

```
<databaseChangeLog>

<include file="10-create-ship-employee-view.sql"
      relativeToChangelogFile="true"/>

</databaseChangeLog>
```

Joonis 26. Chengelog.xml faili näide

4.2.2 Klassid, repositoorium, mapperid

Kuna VIEW-le saab samamoodi juurde pääseda kui tavalisele tabelile, saab selle realiseerida klassina ja seejärel manipuleerida andmetega kui valmis laevadena. Selle tulemusena loodi ülesande lahendamiseks kolm uut klassi:

1. ShipEmployee. See on VIEW päringu tulemus, mis konverteeritakse selle klassi
2. ShipEmployeeId. Lisaklass ShipEmployee tingimuspõhiseks ID-ks.
3. EmployeeAccessCredentialInfo. Klass pääsuvahendite eraldi konverteerimiseks ShipEmployee'ist. Järgnevalt on toodud selle klassi näide annotatsioonidega @Getter, @Setter, @EqualsAndHashCode. [30] [31]

```
@Getter
@Setter
@EqualsAndHashCode
public class EmployeeAccessCredentialInfo {
      private UUID id;
      private String credentialNumber;
      private AccessCardSystemType accessType;}
```

Joonis 27. Töötaja pääsuvahendi info klassi näidis

Lõpuks kogu teave sisaldub kolmes klassides. Projektis asuvad repositooriumid on keskendunud konkreetsetele andmetele, millega nad töötavad. Nii et uue klassi ShipEmployee jaoks luuakse eraldi repositoorium ShipEmployeeRepository, mis kasutab DefaultJpaRepository. Kasutades annotatsiooni @Query, saab otsingu päringuid deklareerida otse repositooriumi meetodis. Kogu filtreerimine, sorteerimine ja võimalik tulevane pagineerimine tehakse kohe repositooriumi meetodis. Otsingu päringu koostamine võib muuta meetodi visuaalselt mahukaks. Näide sellisest päringust asub Lisa 3. Kasutatud DTO sisaldab samu välju nagu põhiklass. [32] [33]

Muutujate konverteerimiseks kasutatakse annotatsiooniga @Mapper liidest. [34]

```
@Mapper(uses = { EmployeeAccessCredentialInfo.class,
                 AccessCredentialDtoMapper.class})
public interface ShipEmployeeDtoMapper {

  @Mapping(target = "valueToIgnore", ignore = true)
  ShipEmployeeDto toDto(ShipEmployee value);
}
```

Joonis 28. Mapingu meetodi näide

4.2.3 Kontroller, Teenus

Kõigi saadaolevate LT töötajate nimekirja päring asub REST-kontrolleris (*Representational State Transfer* – veebiteenusega suhtlemise liidese arhitektuur). HTTP (*Hypertext Transfer Protocol* – edastusprotokoll) GET-päringuga meetodi märkimiseks kasutatakse annotatsiooni @GetMapping, kus määratakse päringu URL (*Uniform Resource Locator* – veebiaadress). Meetod küsib andmeid teenuselt, teisendab need mudeliks samamoodi nagu DTO ja tagastab tulemuse. [35]

Teenuses toimiv meetod töötab järgmiselt:

1. Autoriseerimise kontroll – kasutaja saab üldse LT töötajate nimekirja pärida

2. Filtri andmete õigsuse kontroll. Kui kasutaja valib tööandja filtri, saadetakse päring kõikidele pääsuvahenditele, mis on temaga seotud. Neid saab ka andmete filtreerimiseks kasutada, kuid kui filtri jaoks saadetakse pääsuvahend, mis ei kuulu valitud tööandjale või kui pääsuvahendid on valitud, kuid tööandjat pole, siis on andmed sobimatud.

3. Tööandjate nimekirja päring, kelle andmeid kasutaja võib vaadata. Algselt küsitakse kõikide LT-ga seotud töötajate nimekirja. See nimekiri filtreeritakse ja tulemuseks on sobivate tööandjate ID-d.

4. Töötajate nimekirja päring filtreerimisega ja tulemuse teisendamine DTO-ks. Iga töötaja riigi koodi asemel näidatakse, kas see kuulub Schengeni riikidesse või mitte.

Lahenduse tulemusena tehakse andmebaasi täiendavaid päringuid pääsu kontrollimiseks ja tööandjate ning nende allhankija kirjete nimekirjade koostamiseks, kuid kogu töötajate teave grupeeritakse kiiresti ja ühes kohas.

```
public List<ShipEmployeeDto> getAvailableEmployees (
      UUID shipId, SearchFilterDto filter) {

authorizationService.checkEmployeesAccess(shipId);
validateFilter(filter);

List<EmploymentChain> employmentChains =
      employmentChainRepository.findByShipAndEmployer(
      shipId, filter.getEmployerId());

List<UUID> employerIds =
      employmentChains.stream()
      .filter(authorizationService::hasEmployeeAccess)
      .map(chain -> chain.getEmployer().getId())
      .toList();

return getValidByFilter(shipId, employerIds, filter).stream()
      .map(shipEmployeeDtoMapper::toDto)
      .toList();
}
```

Joonis 29. Töötaja LT-l teenuse otsingu meetodi näide

4.2.4 Autoriseerimine

Esimene kontrollmeetod, mis töö meetodi alguses läbi viiakse, on autoriseerimine. Autoriseerimise meetodi eesmärk on kontrollida, kas kasutajal on pääs valitud LT-se töötajate nimekirja päringule. Praeguses etapis ei ole oluline, kas kasutajal on vähemalt üks kättesaadav töötaja või tööandja, sest meetod kontrollib ainult kasutaja õige rolli olemasolu.

Kasutajal on pääs LT-se töötajate nimekirjale, kui ta on PTV, KÜ või allhankija. Rollide olemasolu kontrollimiseks tehakse vastav päring andmebaasi ja kontrollitakse kasutaja isikukoodi ja riigi

vastavust. Kui on olemas juriidiline või füüsiline isik PTV rolliga valitud laeva raames, kellel on sama riigi kood ja isikukood nagu kasutajal, tähendab see, et kasutaja ongi PTV.

Tööandja saab muuta oma töötajate pääsu LT-le. ShipEmployee objekti teisendamise ajal DTO-sse saadetakse päring, mis kontrollib, kas valitud töötaja isikukood ja riigi kood kattuvad kasutaja omaga. Kui jah, siis tähendab see, et kasutaja on tööandja ja võib muuta töötaja pääsu.

5 Testimine

Selle töö raames valitud lahenduse tõhususe analüüsimiseks arvutati andmete mahtu ja viidi läbi andmepäringute ajakatsetus. See võimaldas näha, millise kiirusega funktsionaalsus töötaks ümber kujundatud andmebaasi tingimustes. See annab ka aimu, kas lahendus, mis kasutab VIEW andmete dubleerimise grupeerimiseks, on kiiruse osas halvem.

5.1 Andmebaasi versioonid

Testimiseks on loodud kaks andmebaasi - ümber kujundatud versioon ja VIEW versioon. Mõlemad asuvad Docker desktopi eraldi konteinerites. Andmete täitmisel kasutati esimese juhuslikult sorteeritud loendi esimest muutujat, et luua sidemeid väljade vahel. Lõpptulemusena on mõlema andmebaasi algandmed järgmised: [18]

- Laevade arv – 10
- Töötajate arv – 20
- Tööandjate arv – 10
- Pääsuvahendi gruppide arv – 10

5.2 Võimalik andmemaht

Algse testmahuga oli arvestatud, et igas LT-s osaleb umbes 10 tööandjat, kellest igaüks võib olla tööahelas kaks korda. Igal tööandjal on umbes 4 töötajat, seega on tabelis *employer_employee* kokku 40 seost, ja need 4 töötajat osalevad igaüks 10 LT-s, kus on tööandja.

Seega osaleb ühes LT-s 40 töötajat. Kui iga tööandja osaleb LT kaks korda, siis ahela suurus on 20. Kui LT-s osaleb 40 töötajat, kus tööandja on ahelas kaks korda, siis vana versiooni tabelis töötajate kirjeid ühe laeva kohta on 80, samal ajal kui uus versioon sisaldab 40 kirjet. Halvimal

juhul on üks tabel kaks korda suurem kui teine. Mida rohkem on tööandjaid või mida väiksem on LT tööahela suurus, seda väiksem on erinevus kahe versiooni vahel. Kui kõik tööandjad ilmuvad kõigis ahelates ainult üks kord, on mõlemal LT-ga seotud töötajate tabelil sama arv kirjeid.

5.3 Päringu kiiruse testimine

Testimise raames kaaluti võimalust, kus *employment_chain_employee* tabel on andmete topeltkirjetena täpselt kaks korda suurem kui *employer_employee_on_ship* tabel. Võtmed tabelite vaheliste seoste loomiseks valiti algsetest andmetest juhuslikult sorteeritud nimekirjast. Seega on laevade ja tööandjate seoste suurus järgmine:

- Laeval on 12 kuni 38 allhankijate kirjet
- Tööandjal on 16 kuni 26 allhankijate kirjet
- Tööandjal on 2 kuni 8 töötajad
- Töötajal on 1 kuni 3 tööandjaga seost

Kui tabelite tööhõive *employment_chain_employee* ja *employer_employee_on_ship* suhe on 2:1, siis valiti testimahuks 800 ja 400 kirjet

Täitmise aja nägemiseks on kasutatud EXPLAIN ANALYZE lause. [36]

Kõik kirjed VIEW-s olid algusest peale filtritud kuupäevade järgi, seega olid kõik tagastatud kirjed aktiivsed. Kõigi päringute jaoks kasutatakse kuupäeva filtrit, et tulemuste arv oleks sama.

Lause 1. Kõik olemasolevad kirjed tabelis

Lause 2. Töötaja eesnimi järgi filtreerimine

Lause 3. Töötaja ees- ja perekonnanimi järgi filtreerimine

Lause 4. Laeva nimetuse järgi filtreerimine

Lause 5. Töötaja ees- ja perekonnanimi ja laeva nimetuse järgi filtreerimine

Tabel 2. Andmebaasi päringute aeg

	employer_employee_on_ship	v_ship_employee	Tulemus
Lause 1	5-8ms	65-75ms	262 rida
Lause 2	5-7ms	20-24ms	55 rida
Lause 3	3-5ms	8-11ms	16 rida
Lause 4	7-10ms	20-24ms	56 rida
Lause 5	8-11ms	7-10ms	11 rida

Tavalise päringuga aktiivsete kirjete puhul jääb VIEW muudetud andmebaasiga võrreldes alla tabelile, sest koos tavalise päringuga luuakse palju teisi andmeid filtreerivaid ja koguvaid päringuid. Hetkel, mil päring muutub piisavalt keeruliseks ja nõuab teiste tabelite ühendamist, suureneb tööaeg tavalise tabeli puhul, samas päringu aeg väheneb VIEW puhul, kuna kõik vajalikud täiendavad tabelid on juba filtreerimise ajal kaasatud. Sellest võib järeldada, et tavalise päringu korral LT-l töötajate nimekirja kohta VIEW on alati kiiruse mõttes halvem kui muudetud andmebaas. Kuid kui kasutaja kasutab filtreerimist aktiivselt, võib kiirus tasakaalustuda.

6 Tulemused

Valitud lahendus vastab oodatud nõuetele ja lahendab püstitatud ülesande. Allhankija detailandmete lehel, töötajate nimekirjas LT-se protsessis ja üldises nimekirjas ei ole kirjete dubleerimist. Kasutaja näeb ühtset ja ajakohast pilti ning ei ole oluline kui palju alhankija kirjeid tööandjal ühe laeva raames on. Virtuaaltabel grupeerib hästi andmete dubleerimised ja töötab kiiresti filtri, sorteerimise ja lehekülje pagineerimisega.

Töötaja lisamisel üldisesse töötajate nimekirja lisatakse ta automaatselt kõigile LT-dele, kus tööandja on töötajatele ligipääsu avanud. Igal töötaja kirjel on staatus, mis vastab allhankija kirje staatusele. Töötajate staatusi arvutatakse ümber kui neid muudetakse. Seega, kui töötajal on aktiivne pääs LT-le, siis ei ole oluline kui mitu allhankija kirjet aktuaalsed on. Kui vähemalt üks kirje on olemas, siis kasutaja näeb lehel, et pääs on olemas.

Viga kliendirakenduses on parandatud, serverirakendus arvutab dünaamiliselt töötajate kirjete staatusi LT-l ning funktsionaalsuses enam andmeid ei desünkroniseerita. See lahendab täielikult rakenduses esinenud kriitilise probleemi ning on universaalne sarnaste probleemide lahendamisel.

7 Kokkuvõte

Selle lõputöö käigus lahendati LT-se probleem, kus töötajate kirjed kordusid. Andmete desünkroniseerimine põhjustas rakenduses vigu nii kliendi- kui ka serverirakenduse poolt.

Töö analüüs annab ülevaate töö nõuetest ja projektis kasutatavatest tehnoloogiatest, samuti analüüsitakse üksikasjalikult võimalikke probleemide lahendamise võimalusi ja äriprotsesside ja andmebaasi muudatusi, mida tuleb nende realiseerimiseks rakendada. Nende eelistest ja puudustest lähtudes valiti virtuaalsete tabelite kasutamise lahendus. See täielikult vastab loetletud nõuetele ja osutus keerulises olukorras kõige sobivamaks lahenduseks. Töö lõpposas käsitletakse valitud probleemilahenduse rakendamist ja tehti järeldused andmebaasi päringute kiiruse põhjal.

Valitud lahendus vastas ülaltoodud nõuetele ja seda võib pidada edukaks. Kliendirakendusel korduvatest kirjetest õnnestus vabaneda ning serverirakenduse loogika filtreerib neid. Seda tööd võib vaadelda sarnaste probleemide lahendamise näitena või kasutada juba olemasolevaid tulemusi ja järeldusi paremate lahenduste otsimiseks analüüsi algfaasis. Lahendust saab nimetada universaalseks, kuna paljud andmebaasid toetavad virtuaaltabelite loomist. Selle tabeli loomine on jagatud andmeplokkideks ja sellesse saab kergesti muudatusi teha. Sellise lahenduse kasutamine tõmbab alguses tabelist välja ainult tulevikus vajalikud väljad, mis vähendab andmete töötlemise aega ja töö toimub kohe vajalike väljadega.

Kasutatud kirjandus

[1] Atlassian, „What is Agile?,“ [Võrgumaterjal]. Available: https://www.atlassian.com/agile. [Kasutatud 10 aprill 2023].

[2] S. Knipe, „Data Trees as a Means of Presenting Complex Data Analysis,“ Community of Knowledge, 8 august 2013. [Võrgumaterjal]. Available: https://www.community-of-knowledge.de/beitrag/data-trees-as-a-means-of-presenting-complex-data-analysis/. [Kasutatud 10 aprill 2023].

[3] K. NAVE, „Set sail to the bonkers birthplace of the world's largest container ships,“ WIRED, 16 september 2014. [Võrgumaterjal]. Available: https://www.wired.co.uk/gallery/ship-gallery. [Kasutatud 10 aprill 2023].

[4] „Functional vs Non-functional Requirements,“ Visure Solutions, Inc., [Võrgumaterjal]. Available: https://visuresolutions.com/requirements-management-traceability-guide/functional-vs-non-functional-requirements. [Kasutatud 10 aprill 2023].

[5] „Spring Framework Overview,“ VMware Inc., 20 märts 2023. [Võrgumaterjal]. Available: https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html. [Kasutatud 10 aprill 2023].

[6] „ASP.NET overview,“ Microsoft, 30 september 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/overview. [Kasutatud 10 aprill 2023].

[7] H. Dhaduk, „Angular vs React: Which to Choose for Your Front End in 2023?,“ Simform, 6 aprill 2023. [Võrgumaterjal]. Available: https://www.simform.com/blog/angular-vs-react/#:~:text=How%20is%20React%20different%20from,has%20a%20smaller%20bundle%20size.. [Kasutatud 10 aprill 2023].

[8] A. Phaujdar, „PostgreSQL vs Oracle: 6 Critical Differences,“ Hevo Data Inc., 2 november 2021. [Võrgumaterjal]. Available: https://hevodata.com/learn/postgresql-vs-oracle/. [Kasutatud 25 veebruar 2023].

[9] P. Tolmachev, „Первое изменение исходного кода PostgreSQL,“ 19 detsember 2021. [Võrgumaterjal]. Available: https://ptolmachev.ru/pervoe-izmenenie-isxodnogo-koda-postgresql.html. [Kasutatud 25 veebruar 2023].

[10] M. Smallcombe, „PostgreSQL vs MySQL:The Critical Differences,“ Integrate.io, 15 veebruar 2023. [Võrgumaterjal]. Available: https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/#:~:text=The%20Critical%20Differences%20of%20Postgres%20vs%20MySQL%3A&text=MySQL%20is%20a%20simpler%20database,pure%20relational%20database%20(RDBMS).. [Kasutatud 10 aprill 2023].

[11] „JSON in Oracle Database,“ Oracle, [Võrgumaterjal]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/21/adjsn/json-in-oracle-

database.html#GUID-D7BCE045-EF6D-47E9-9BB2-30C01933248E. [Kasutatud 10 aprill 2023].

[12] „Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others," AltexSoft, 27 oktoober 2021. [Võrgumaterjal]. Available: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/. [Kasutatud 2023 aprill 10].

[13] C. McKenzie, „Tomcat vs. Jetty: How these Java application servers compare and differ," TechTarget, 26 november 2019. [Võrgumaterjal]. Available: https://www.theserverside.com/video/Tomcat-vs-Jetty-How-these-application-servers-compare#:~:text=Tomcat%20is%20an%20Apache%20project,the%20Eclipse%20Public%20License%201.0.. [Kasutatud 1 märts 2023].

[14] „GlassFish vs Tomcat – What's the Difference between Tomcat & GlassFish?," DailyRazor, [Võrgumaterjal]. Available: https://www.dailyrazor.com/blog/glassfish-vs-tomcat/. [Kasutatud 10 aprill 2023].

[15] „Developer Guide," WildFly , 14 detsember 2021. [Võrgumaterjal]. Available: https://docs.wildfly.org/26/Developer_Guide.html. [Kasutatud 2023 aprill 10].

[16] „Lucidchart Release Updates," Lucid Software Inc, 2023. [Võrgumaterjal]. Available: https://lucidchart.zendesk.com/hc/en-us/articles/360031317151-Release-Updates?gclid=CjwKCAjw586hBhBrEiwAQYEnHZmzj4rEMliga9_UqlIs_T0YglbSsaeuliyxXNj21qt2FNvZtVAk7xoCP1YQAvD_BwE&km_CPC_AdGroupID=115189963814&km_CPC_AdPosition=&km_CPC_CampaignId=12085501864&. [Kasutatud 8 aprill 2023].

[17] „Release Notes," Figma, [Võrgumaterjal]. Available: https://www.figma.com/release-notes/. [Kasutatud 8 aprill 2023].

[18] „Containerize an application," Docker Inc., [Võrgumaterjal]. Available: https://docs.docker.com/get-started/02_our_app/. [Kasutatud 16 veebruar 2023].

[19] „What is IntelliJ IDEA?," JetBrains s.r.o., [Võrgumaterjal]. Available: https://www.jetbrains.com/idea/features/. [Kasutatud 28 jaanuar 2023].

[20] „Intellij vs VSCode – Which is best for Java devs?," Tabnine, 4 november 2019. [Võrgumaterjal]. Available: https://www.tabnine.com/blog/intellij-vs-vscode/. [Kasutatud 20 veebruar 2023].

[21] „IntelliJ vs Eclipse: Which is better for beginners?," Great Lakes E-Learning Services Pvt. Ltd., 22 november 2022. [Võrgumaterjal]. Available: https://www.mygreatlearning.com/blog/intellij-vs-eclipse/. [Kasutatud 20 veebruar 2023].

[22] „IntelliJ vs Eclipse," InterviewBit, 5 oktoober 2021. [Võrgumaterjal]. Available: https://www.interviewbit.com/blog/intellij-vs-eclipse/. [Kasutatud 20 vaabruar 2023].

[23] „Представления," Microsoft, 3 märts 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/ru-ru/sql/relational-databases/views/views?view=sql-server-ver16. [Kasutatud 21 märts 2023].

[24] „CREATE VIEW (Transact-SQL)," Microsoft, 25 jaanuar 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/ru-ru/sql/t-sql/statements/create-view-transact-sql?view=sql-server-ver16. [Kasutatud 21 märts 2023].

[25] В. Лукьянчиков, „Представления (VIEW) в MySQL,“ SQLinfo.ru, 14 detsember 2008. [Võrgumaterjal]. Available: https://sqlinfo.ru/articles/info/9.html. [Kasutatud 21 märts 2023].

[26] „Schengen Area – The World’s Largest Visa Free Zone,“ SchengenVisaInfo.com, [Võrgumaterjal]. Available: https://www.schengenvisainfo.com/schengen-visa-countries-list/. [Kasutatud 10 aprill 2023].

[27] „Changelog,“ Liquibase Inc, [Võrgumaterjal]. Available: https://docs.liquibase.com/concepts/changelogs/home.html. [Kasutatud 20 märts 2023].

[28] „Example Changelogs: SQL Format,“ Liquibase Inc, [Võrgumaterjal]. Available: https://docs.liquibase.com/concepts/changelogs/sql-format.html. [Kasutatud 20 märts 2023].

[29] „Changeset,“ Liquibase Inc, [Võrgumaterjal]. Available: https://docs.liquibase.com/concepts/changelogs/changeset.html. [Kasutatud 20 märts 2023].

[30] „@Getter and @Setter,“ The Project Lombok Authors, [Võrgumaterjal]. Available: https://projectlombok.org/features/GetterSetter. [Kasutatud 20 märts 2023].

[31] „@EqualsAndHashCode,“ The Project Lombok Authors, [Võrgumaterjal]. Available: https://projectlombok.org/features/EqualsAndHashCode. [Kasutatud 20 märts 2023].

[32] O. Gierke, C. Strobl, M. Paluch, S. Krabbenborg, J. Wouters, G. Turnquist ja J. Schauder, „Interface JpaRepository<T,ID>,“ Pivotal Software, Inc., [Võrgumaterjal]. Available: https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html. [Kasutatud 20 märts 2023].

[33] O. Gierke, T. Darimont, C. Strobl, M. Paluch, J. Bryant ja G. Turnquist, „Spring Data JPA - Reference Documentation,“ Pivotal Software, Inc., 3 märts 2023. [Võrgumaterjal]. Available: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repository-query-keywords. [Kasutatud 20 märts 2023].

[34] G. Morling, A. Gudian, S. Derksen ja F. Hrisafov, „MapStruct 1.5.3.Final Reference Guide,“ The MapStruct authors, 7 oktoober 2022. [Võrgumaterjal]. Available: https://mapstruct.org/documentation/stable/reference/html/#defining-mapper. [Kasutatud 23 märts 2022].

[35] S. Brannen, „Annotation Interface GetMapping,“ Spring Framework, [Võrgumaterjal]. Available: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/GetMapping.html. [Kasutatud 23 märts 2023].

[36] „EXPLAIN,“ The PostgreSQL Global Development Group, [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/sql-explain.html#:~:text=The%20ANALYZE%20option%20causes%20the,of%20rows%20it%20actually%20returned.. [Kasutatud 23 märts 2023].

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

Mina, Ekaterina Afanasjeva

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Korduvate klientide leidmise lahendus", 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.

10.04.2023

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 – View tabeli loomine

```
create view v_ship_employee as
select distinct employee_id,
                employee_first_name,
                employee_last_name,
                employee_person_code,
                employee_citizenship,
                employee_residency,
                employee_date_of_birth,
                employee_document_number,
                employer_id,
                employer_name,
                employer_identification_code,
                access_credentials,
                employer_employee_id,
                employer_employee_relation_type,
                employer_employee_data_source,
                ship_access,
                ship_id,
                ship_name,
                ship_type,
                ship_imo_number,
                ship_call_sign,
                ship_flag_registry
```

Joonis 30. Tabeli väljade koostamine

```
from  (with employment_chain_employees as
      (select ec.id                     as employment_chain_id,
              s.id                      as ship_id,
              s.name                    as ship_name,
              s.type                    as ship_type,
              s.imo_number              as ship_imo_number,
              s.call_sign               as ship_call_sign,
              s.flag_registry           as ship_flag_registry,
              ece.employer_employee_id  as employer_employee_id,
              ece.valid_from            as ece_valid_from,
              ece.valid_to              as ece_valid_to
      from employment_chain ec
            left join employment_chain_employee ece
                  on ec.id = ece.employment_chain_id
            left join ship s
                  on s.id = ec.ship_id)
```

Joonis 31. Põhine subselect tabeli koostamisel. Kombinatsioon laev – töövõtuahel – töötaja töövõtuahelas

```
select e.id                      as employee_id,
       e.first_name              as employee_first_name,
       e.last_name               as employee_last_name,
       e.person_code             as employee_person_code,
       e.citizenship             as employee_citizenship,
       e.residency               as employee_residency,
       e.date_of_birth           as employee_date_of_birth,
       e.document_number         as employee_document_number,

       er.id                     as employer_id,
       er.name                   as employer_name,
       er.identification_code    as employer_identification_code,

       ee.id                     as employer_employee_id,
       ee.relation_type          as employer_employee_relation_type,
       ee.data_source            as employer_employee_data_source,

       ece.ship_id,
       ece.ship_name,
       ece.ship_type,
       ece.ship_imo_number,
       ece.ship_call_sign,
       ece.ship_flag_registry
```

Joonis 32. Töötaja, tööandja ja laeva põhiandmete tabelisse kirjutamine

```
(select jsonb_agg(credential_info)
from  (select jsonb_build_object(
            'id', ac.id,
            'accessType', ac.access_type,
            'credentialNumber', ac.credential_number)
      as credential_info
      from employer_employee see
            inner join access_credential ac
      on see.access_credential_list_id = ac.access_credential_list_id
      where see.id = ee.id
      and ac.valid_from is not null
      and ac.valid_from <= current_date
      and (ac.valid_to is null or ac.valid_to > current_date)
      union
      select jsonb_build_object(
            'id', ac.id,
            'accessType', ac.access_type,
            'credentialNumber', ac.credential_number)
      as credential_info
      from employer_employee see
            inner join employee e on e.id = see.employee_id
            inner join access_credential ac
      on e.access_credential_list_id = ac.access_credential_list_id
      where see.id = ee.id
      and ac.valid_from is not null
      and ac.valid_from <= current_date
      and (ac.valid_to is null or ac.valid_to > current_date))
            as access_credentials)
      as access_credentials,
```

Joonis 33. Subselect – töötaja pääsuvahendi võtmine

```
(select case
      when exists(select 1 from employment_chain_employees
                  where employer_employee_id = ee.id
                  and ship_id = ece.ship_id
                  and (ece_valid_from is null
                       or ece_valid_from <= current_date)
                  and (ece_valid_to is null
                       or ece_valid_to > current_date))
      then true
      else false end)
as ship_access,
```

Joonis 34. Subselect – LT pääsu otsimine

```
from employer_employee ee
      left join employee e on e.id = ee.employee_id
      left join employer er on er.id = ee.employer_id
            inner join employment_chain_employees ece
      on ee.id = ece.employer_employee_id
where (ee.valid_from is null or ee.valid_from <= current_date)
and (ee.valid_to is null or ee.valid_to > current_date))
as ship_employee;
```

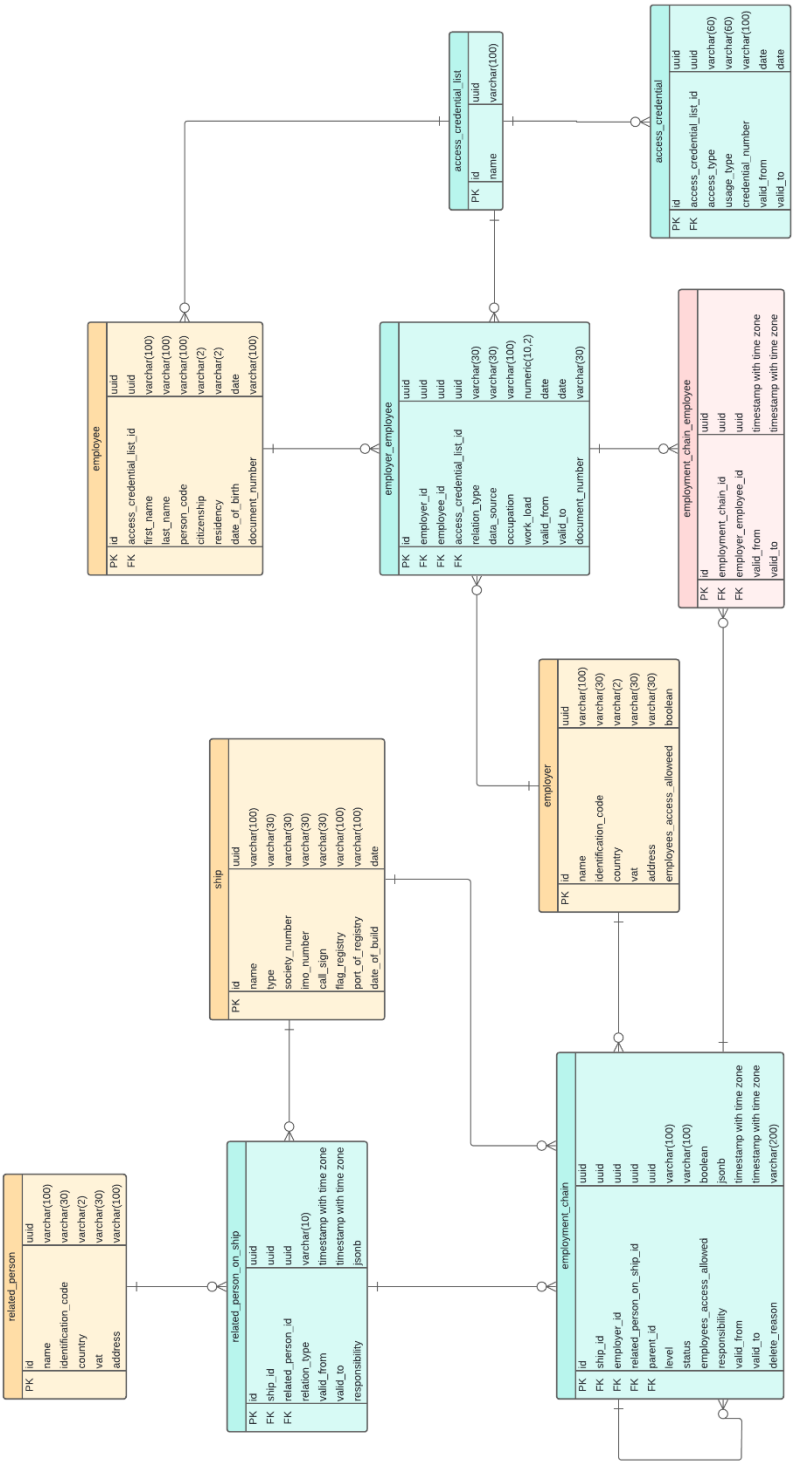
Joonis 35. Põhi olemuse lisamine

Lisa 3 – Töötaja laeva tootmisel filtreerimine

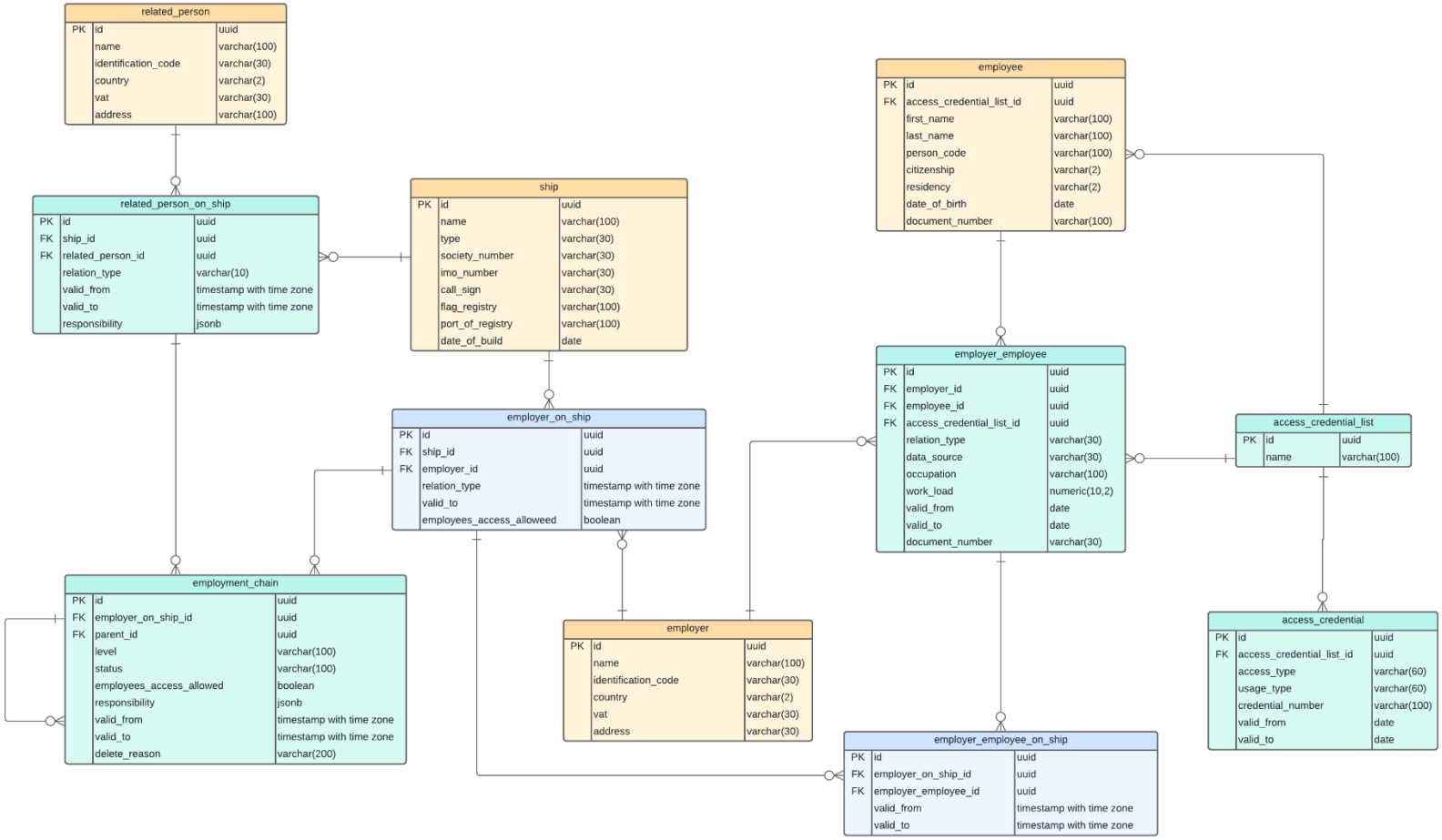
```
@Query("""
SELECT shipEmployee FROM ShipEmployee shipEmployee
WHERE shipEmployee.shipId = :shipId
AND shipEmployee.employerId IN :employerIds


AND (:employee IS NULL OR UPPER(shipEmployee.employeePersonCode)
LIKE CONCAT('%', :employee, '%')
AND (:hasAccess IS NULL OR shipEmployee.shipAccess = :hasAccess)


AND (:schengenResidency IS NULL
OR (:schengenResidency = TRUE
    AND shipEmployee.employeeResidency IN (:schengenCountries))
OR (:schengenResidency = FALSE
    AND shipEmployee.employeeResidency NOT IN (:schengenCountries)))
    """)
Page<ConstructionEmployee> findAllBySearchParams(
     @Param("shipId ") UUID shipId,
     @Param("employerIds ") List<UUID> employerIds,
     @Param("employee") String employee,
     @Param("hasAccess") Boolean hasAccess,
     @Param("shengenResidency ") Boolean shengenResidency,
     @Param("schengenCountries") List<CountryCode> schengenCountries
);
```

Joonis 36. Töötaja LT-l filtri meetodi näide

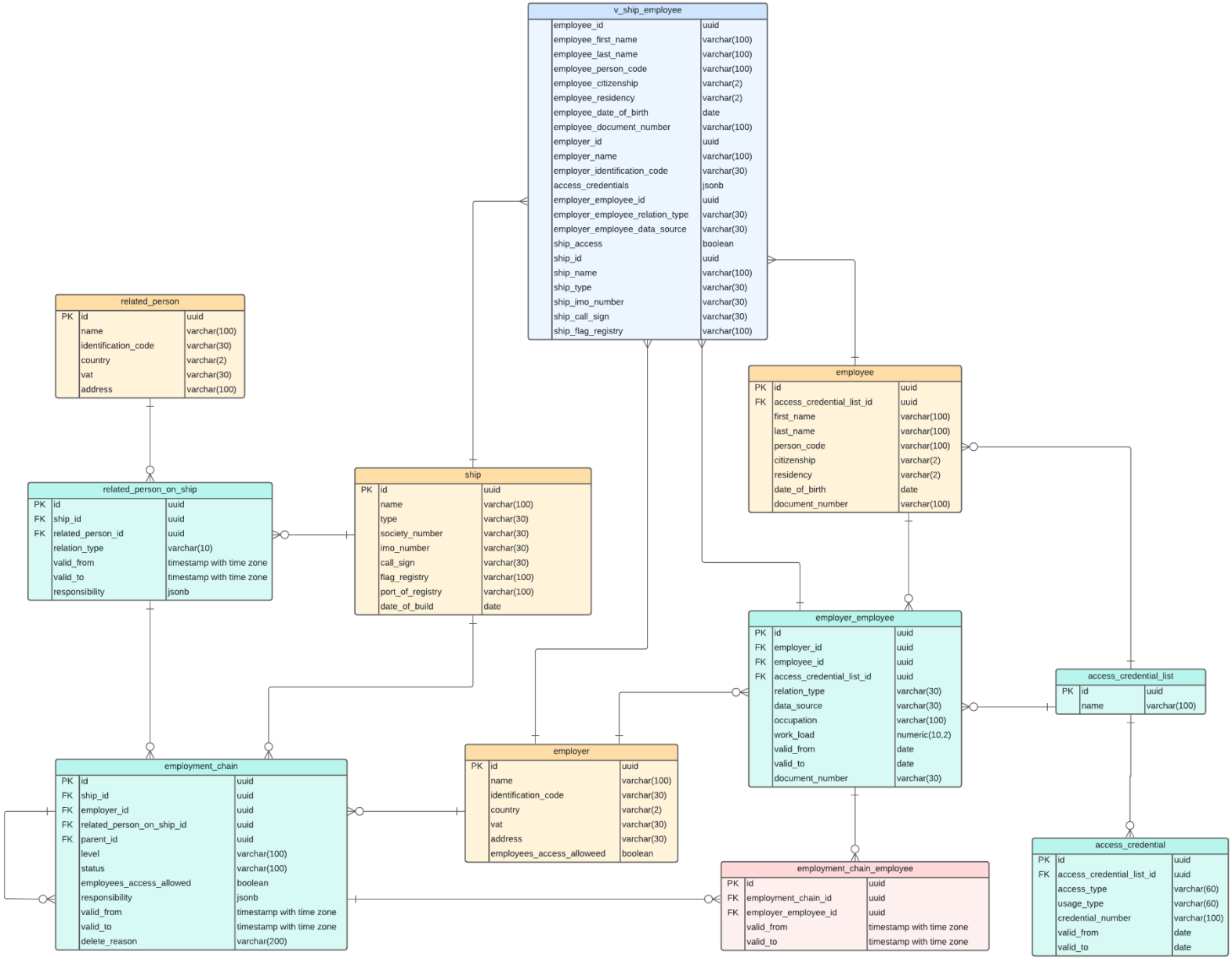
Lisa 4 – EDR mudelid



Joonis 37. Andmebaasi ERD mudel. Vana funktsionaalsus



Joonis 38. Andmebaasi ERD mudel. Andmebaaside ümberkirjutamine



Joonis 39. Andmebaasi ERD mudel. Grupeerimine SQL lausega