TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kerli Ruul 179694IADB

BÖRSIETTEVÕTETE FINANTSANDMETE AUTOMATISEERITUD KAARDISTAMINE

Bakalaureusetöö

Juhendaja: Tõnn Talpsepp
Doktorikraad

Tallinn 2020

Autorideklaratsioon

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

Autor: Kerli Ruul

18.05.2020

Annotatsioon

Antud lõputöö eesmärk on automatiseerida finantsandmete kaardistamist. Eesmärgi täitmiseks lahendab töö autor järgmisi ülesandeid:

- jagab finantsandmete kaardistamise gruppideks;
- loob ühtse struktuuri finantsandmete kaardistamise valemitele;
- loob loogika, mis oskab eelmises punktis loodud struktuuris valemiga finantsandmeid arvutada;
- loob vaate ettevõtte kõikide finantsandmete nägemiseks – vajalik finantsandmete kaardistamiseks;
- loob vaate kasutatud finantsandmete kaardistamise valemitest – kasutajal lihtsam mõista, mis valemit süsteem kasutas finantsandmete arvutamiseks;
- loob võimaluse muuta finantsandmete kaardistamise valemeid kasutajaliideses;
- loob õpetuse finantsandmete kaardistamise valemite loomiseks.

Rakendus, kuhu automatiseeritud finantsandmete kaardistus lahendatakse, on investorite tööriist ettevõtete finantsandmete analüüsimiseks.

Loodav lahendus on vajalik administraatorkasutajatele, et kiirendada ja lihtsustada finantsandmete kaardistamist. Seoses sellega on loodavad vaated nähtav ainult rakenduse administraatoritele. Hetkel on finantsandmete kaardistamine lahendatud koodi tasemel ehk administraatoril pole võimalik finantsandmete kaardistus muuta ilma arendaja sekkumiseta. Idee on arendaja vahelt ära kaotada ja administraatorile luua kasutajaliidesesse vaated, kust saab näha vajaminevat informatsiooni ja vastavalt soovile finantsandmete kaardistust muuta.

Loodavas kasutajaliidese vaates kaardistatakse finantsandmeid valemi kujul ehk administraator paneb kokku valemi vastavalt finantsandmete väljadele. Hiljem kasutatakse neid valemeid finantsnäitajate arvutamisel.

Valemi kujul kaardistamiseks luuakse ühtne struktuur valemite koostamiseks ja loogika valemitega finantsandmete arvutamiseks. Lisaks luuakse õpetus valemite koostamiseks, et muutja teaks milline on valemite struktuur ja võimalikud tehted.

Selleks, et administraatoril oleks võimalik finantsandmeid kaardistada – näha vajaminevat infot, luuakse ka vaated kõikidele finantsandmetele ja eelnevalt kasutatud valemitele.

Töö lõplikuks tulemuseks on administraatoril võimalik näha vaja vajaminevat infot (finantsandmeid, kasutatud valemeid ja valemite koostamise õpetus), et kasutajaliidese vaates iseseisvalt finantsandmeid kaardistada.

Töö funktsionaalsuse valideerimiseks kirjutab lõputöö autor tagasüsteemi ühiktestid ning kontrollib manuaalselt kasutajaliidest, et loodud vaated avaneksid ja näitaksid õiget infot.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 6 peatükki, 14 joonist, 2 tabelit.

Abstract

Automated Mapping of Financial Data of Listed Companies

The aim of this thesis is to automate the mapping of financial data. In order to fulfill the goal, the author solves the following tasks:

- divides financial data mappings into groups;
- creates a unified structure for financial data mapping formulas;
- creates a logic that can calculate financial data with formular structure was created in the previous point;
- creates a view to see all the financial data of the company - necessary for mapping financial data;
- creates a view of used financial data mapping formulas - it is easier for the user to understand which mapping formula was used by the system;
- creates an opportunity to change financial data mapping formulas in the user interface;
- creates a tutorial on creating financial data mapping formulas.

An application for automated financial data mapping is an investor's tool for analyzing corporate financial data.

Created views are necessary for administrator users to speed up and simplify financial data mapping. Created views are visible only for administrators. At the moment, automated financial data mapping is solved at the code level and the administrator can’t change financial data mapping without developers. The idea is to eliminate developers on this flow and create an user interface for the administrator, where administrator can change financial data mapping as desired.

Administrator will map financial data in formula form – create formula based on financial data fields. These formulas are later used to calculate financial indicators.

To map financial formulas it's needed to create unified structure for formulas and logic for calculating it with financial data. In addition, a tutorials are provided on how to create a financial data mapping formulas with possible formula operations and examples of formula compilation.

In order for administrator to be able to map financial data, to see necessary information, author will create all financial data view and used formulas view.

Outcome of this thesis is that the administrator can see all the necessary information (financial data, used mapping formulas and tutorial on how to make the mapping formulas) in order to map financial data independently in the user interface.

To validate solution functionality author writes unit tests in backend system and checks manually user interface, so that the created user interface displays the correct information and works properly.

The thesis is in Estonian and contains 33 pages of text, 6 chapters, 14 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	Inglise keeles *Application Programming Interface*. Arvutiprogrammides alamprogrammi määratluste, protokollide ja tööriistade komplekt rakendustarkvara ehitamiseks.
endpoint	Lõpp-punkt URL kujul, mis määrab ära ressursside asukoha.
ESMA	Inglise keeles *European Securities and Markets Authority*. Euroopa Väärtpaberiturujärelevalve.
PG	Ettevõtte Procter & Gamble Company börsitähis.
SEC	Inglise keeles *Securities and Exchange Commission*. USA väärtpaberi- ja börsikomisjon.
URL	Inglise keeles *Uniform Resource Locator*. Internetiaadress ehk üldine infoallika asukohamääraja.
USA	Inglise keeles *The United States of America*. Ameerika Ühendriigid.
XBRL	Inglise keeles *eXtensible Business Reporting Language*. XML-põhine infoedastusstandard.
XML	Inglise keeles *Extensible Markup Language* Laiendatav märgistuskeel.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Finantsturud on muutunud ajas aina rohkem automatiseerituks, kus enamik tehinguid teevad arvutid. Seetõttu on arvutitel otsuste tegemisel suur roll finantsandmete analüüsis. Arvutitele paremaks mõistmiseks on vaja andmed kaardistada õigesti. Turul on vajadus omada detailsemaid andmeid ning finantsanalüütikul vajadus andmeid kaardistada vastavalt.

Väga suur ja kasvav rühm seadusandjaid, ettevõtteid, valitsusi ja teabeökosüsteeme on aruanded viinud XBRL-i standardile. XBRL on rahvusvaheline standard digitaalseks aruandluseks. Näiteks finants-, tulemuste-, riski-, nõuete- ja muude aruannete esitamiseks [1].

Tänu XBRL standardile on USA börsiettevõtte aruanded järjepidevad, võrreldavad ja masinloetavad ning seda juba üle 10 aasta. Alates 2008 aastast USA väärtpaberi- ja börsikomisjon (SEC) kohustanud kõiki USA börsiettevõtteid esitama aruandeid XBRL standardis [2]. Euroopa börsiettevõtted pole veel ühtse standardini jõudnud, aga Euroopa Väärtpaberiturujärelevalve (ESMA) võttis vastu seaduse, et 2020 aasta aruanded hakkavad vaiksel üle minema XBRL standardile ja 2022 aastaks on kohustus kõikidel Euroopa börsiettevõtetel esitata aruanded XBRL standardis [3].

Tänu XBRL finantsandmete standardile on aruanded masintöödeldavad ja muudab andmete kogumise kergemaks ja odavamaks [2].

Ainult andmetes ei piisa, kui andmete sisu ei tea. Probleem on selles, et samal andmeväljal võib olla erinev nimi või esitatakse andmeid erineva detailsusastmega. See muudavad andmete edasise analüüsi keerukaks ning andmed pole lihtsalt võrreldavad ega töödeldavad. Lõputöös kajastatavas rakenduses on finantsandmete kaardistamine lahendatud koodis, mille muutmine on tülikas ja aeganõudev. See omakorda tekitab vajaduse kasutajaliidesele, millega saaks erinevaid andmevälju kaardistada, ilma koodi muudatusi tegemata.

Antud lõputöö eesmärk on automatiseerida finantsandmete kaardistamist. Eesmärgi täitmiseks lahendab töö autor järgmised ülesanded:

- jagab finantsandmete kaardistamise gruppideks;
- loob ühtse struktuuri finantsandmete kaardistamise valemitele;
- loob loogika, mis oskab eelmises punktis loodud struktuuris valemiga finantsandmeid arvutada;
- loob vaate ettevõtte kõikide finantsandmete nägemiseks – vajalik finantsandmete kaardistamiseks;
- loob vaate kasutatud finantsandmete kaardistamise valemitest – kasutajal lihtsam mõista, mis valemit süsteem kasutas finantsandmete arvutamiseks;
- loob võimaluse muuta finantsandmete kaardistamise valemeid kasutajaliideses;
- loob õpetuse finantsandmete kaardistamise valemite loomiseks.

Automatiseeritud finantsandmete kaardistamise lahenduse eeldused on järgmised:

- finantsandmete analüüsimise rakendus, kuhu lahendust ehitatakse;
- loogika börsiettevõtete finantsandmete kätte saamiseks;
- loogika finantsandmete töötlemiseks ja arvutamiseks.

Töö on jagatud nelja peatükki:

- probleemi kirjeldus ja nõudmised;
- rakenduse tutvustus;
- finantsandmete automatiseeritud kaardistamise lahendus;
- lahenduse analüüs ja järeldused.

2 Probleemi kirjeldus ja nõudmised

Rakenduses on kahte tüüpi kasutajaid – tavakasutaja ja administraator. Täpsemalt rakenduse erinevatest kasutajatest on juttu järgmises peatükis – rakenduse tutvustuses.

Rakenduses on ettevõtte andmed avalikud, kui seda näevad ka tavakasutajad. Enne seda, kui ettevõtte andmed saavad avalikuks, on administraator suure eeltöö ära teinud. Eeltööks on ettevõtte sisestamine süsteemi, finantsandmete kontrollimine ja parandamine ning avalikustamine.

Administraatori jaoks on kõige suurem ajakulu finantsandmete kontrollimine ja parandamine. Administraatoril on oluline kontrollida finantsnäitajate kaardistamist, sest finantsnäitaja võib olla esitatud algandmetes erineva nime või detailsusastmega. Seose sellega, et antud lõputöö puudutab ainult finantsandmete valemite kaardistamist, siis töö autor pikemalt finantsandmete kontrollimise protsessi ei kirjelda.

Tavaliselt, olukorras, kus administraator avastab kontrollimise käigus, et mingi välja väärtus ei tundu õige, pole tal võimalik seda kontrollida ega parandada järgmistel põhjustel:

- pole võimalik näha, mis loogikat välja väärtuse arvutamisel kasutati;
- pole võimalik näha, mis on kõikide väljade nimed ja väärtused;
- ei pruugi olla teadmisi, et koodi muudatusi teha.

Selleks, et välja arvutamise loogikat ja kõikide väljade nimesid ning väärtusi näha, peab administraator arendajaga suhtlema. Arendaja ei pruugi alati kohe sellega tegeleda ja ühe ettevõtte avalikustamine jääb venima.

Saades kõik vajalikud andmed arendajalt on administraatoril võimalik finantsnäitaja väärtuse õigsust kontrollida. Leides finantsandmete kaardistamises vea, peab administraator jälle arendajaga suhtlema, et ta koodis valemi kaardistamise ära parandaks.

Sellist protsessi peab administraator iga kord läbima, kui ta finantsnäitaja väärtuses kahtleb või on näitaja valesti kaardistatud. See on nii administraatori kui ka arendaja jaoks tülikas ja aeganõudev tegevus.

Lahenduse eesmärk on anda administraatorkasutajale iseseisvus, et tal oleks võimalik andmeid kontrollida ja finantsandmete kaardistust muuta ilma, et peaks kaasama arendajat. Tänu millele kiireneb rakenduses uute ettevõtete avalikuks tulek, sest ei pea enam arendajaga suhtlema ja tema järgi ootama. Lisaks pole vaja arendaja aega kulutada finantsandmete kaardistamise loogika muutmiseks.

Töö autor püstitas järgmised nõuded loodavale lahendusele:

- rakenduses olevate ettevõtete finantsandmed peavad jääma muutumatuks;
- lisatud funktsionaalsused on kaitstud administraatori õigustega;
- administraatoril on võimalik näha kõiki finantsandemete väljade nimesid ja väärtusi;
- administraatoril on võimalik näha kasutatud valemeid;
- administraatoril on võimalik näha valemid kindla ettevõtte finantsnäitaja kohta;
- administraatoril on näha õpetust finantsandmete kaardistamise valemite koostamise kohta;
- administraatoril on võimalik muuta finantsandmete kaardistamist;
- administraator ei pea arendajaga koostööd tegema, et saada vajaminevat infot finantsandmete kaardistamiseks ja valemite muutmiseks.

Selleks, et valideerida lahendust, peab finantsandmete automatiseeritud kaardistamine läbima järgmisi testjuhtumeid:

- on võimalik näha ettevõtte kõiki finantsandmete väljade nimesid ja väärtusi;
- valemite muutmise vaadet avades eeltäidetakse vorm eelnevalt salvestatud valemiga;
- valemeid koostades on võimalik näha juhendit valemite koostamise kohta;

- salvestades ebakorrektse valemi tuleb vastav teade ja valemit ei salvestata;
- salvestades korrektse valem tuleb vastav teade ja valem salvestati;
- peale valemi muutmist ja uuesti finantsandmeid arvutades muutub finantsnäitaja väärtus;
- ettevõtte finantsnäitaja valemit muutes ei muutu teise ettevõtte finantsnäitaja valem;
- valemite vaadet avades on näha kasutatud valemeid finantsnäitajate kaupa.

3 Rakenduse tutvustus

Rakendus on ehitatud selleks, et lihtsustada investoril ettevõtete analüüsimist, andes ülevaate ettevõtte finantstulemustes nii numbriliselt kui ka graafiliselt.

Varasemalt, enne rakenduse olemasolu, analüüsiti ettevõtteid Excelis. Exceli puuduseks oli investorite arvates andmete manuaalne sisestamine, mis oli kasutajate jaoks liiga suur ajakulu. Nii tuli rakenduse autoritel idee ehitada rakendus, mis automatiseerib manuaalase sisestamise. Lõputöö autor on ise ka üks rakenduse autoritest.

Aja jooksul on rakendust arendatud. Näiteks on lisatud portfelli haldusega seotud funktsionaalsust.

Rakendust on jagatud välja ka teistele ja selle tulemusel on rakendusel tekkinud uusi kasutajaid ehk see pole enam ainult rakenduse autorite ettevõtete analüüsimise tööriist.

Hetkel on rakendus kasutajate jaoks ingliskeelne.

Järgmistes alapeatükkides lõputöö autor kirjeldab rakenduse funktsionaalsust, annab ülevaate rakenduses kasutavatest tehnoloogiatest ja konkureerivatest rakendustest.

3.1 Rakenduse funktsionaalsus

Rakenduse eesmärk on lihtsustada investoril ettevõtte analüüsimist ja pakkuda vajalikku funktsionaalsust ühes kohast.

Antud rakendus on konto põhine ehk kasutamiseks on vaja omada registreeritud kasutajat. Rakenduses on kahte tüüpi kasutajaid:

- tavakasutaja;
- administraator;

Tavalisel ja administraator tüüpi kasutajal on rakenduses erinevad õigused. Seetõttu on neil näha ka erinevad vaated. Tabel 1 on võimalik näha nimekirja rakenduse vaadetest ja näha kasutajate õigusi vaadete kaupa.

Kasutajakonto omamine muudab ka rakenduse kasutamise kasutaja jaoks mugavamaks. Näiteks saab investor lisada märkmeid ettevõtte kohta, lisada ettevõtteid vaatlusnimekirja ja kasutada portfelli haldust.

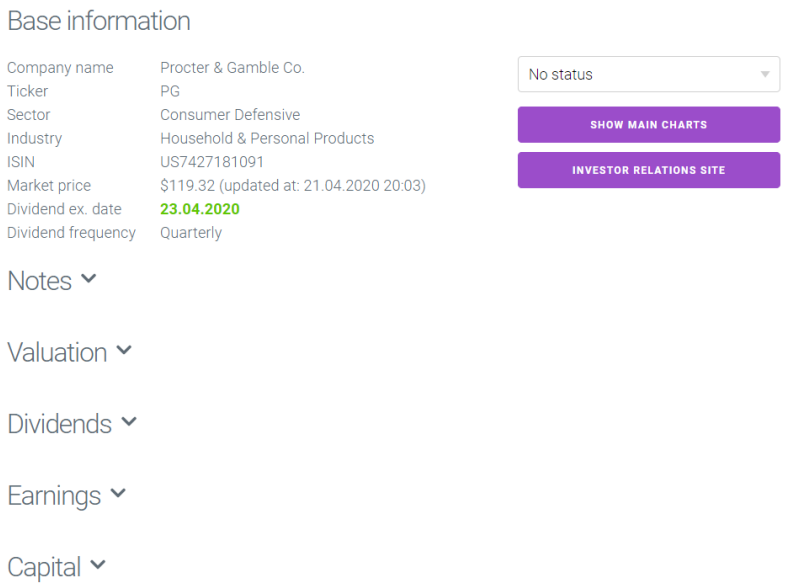
Tabel 1. Rakenduse vaated ja kasutajate õigused vaadete kaupa.

Vaade	**Õigused**	**Kasutajagrupp**
Avalikustatud ettevõtete nimekiri	Näha avalikustatud ettevõtete nimekirja.	Tavakasutaja ja administraator
Vaatlusnimekiri	Näha vaatlusnimekirja lisatud ettevõtteid.	Tavakasutaja ja administraator
Ettevõtte detailvaade	Avalikustada ettevõtte.	Administraator
	Näha ettevõtte finantsandmeid ja suhtarve.	Tavakasutaja ja administraator
	Lisada ettevõtte kohta märkmeid.	Tavakasutaja ja administraator
	Lisada ettevõtte vaatlusnimekirja.	Tavakasutaja ja administraator
Kannete vaade	Võimalik lisada ostu ja müügi kandeid.	Tavakasutaja ja administraator
Portfelli ülevaade	Vastaval kannetel on võimalik saada ülevaadet portfellist.	Tavakasutaja ja administraator
Dividendide ülevaade	Näha ülevaadet tulevastest dividendidest ja ühe ettevõtte dividendide osakaalu tervikust.	Tavakasutaja ja administraator
Uue ettevõtte lisamise vaade	Võimalik lisada uus ettevõtete rakendusse.	Administraator
Avalikustamata ettevõtete nimekiri	Näha kõiki rakendusse lisatud ettevõtteid, mis ei ole veel avalikustatud.	Administraator

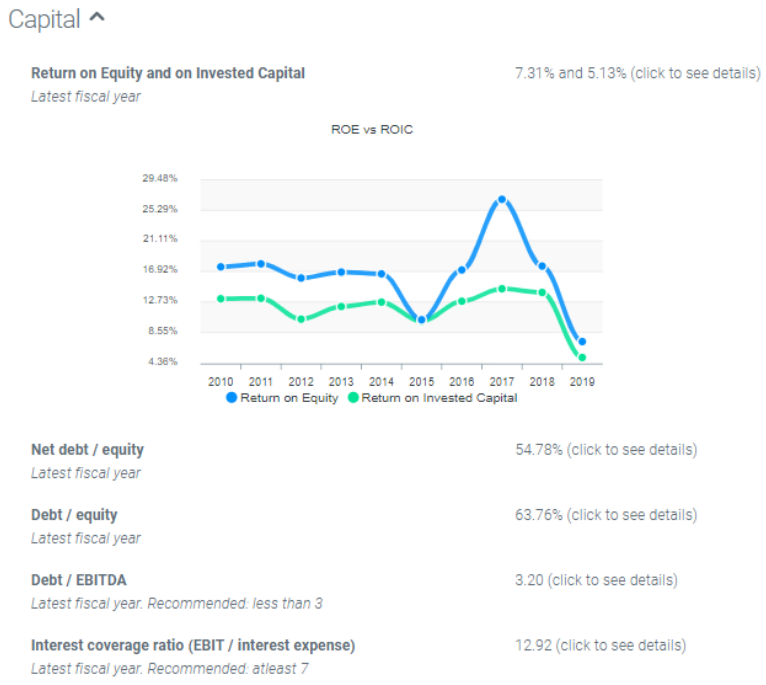
Tabel 1 loetletud vaadetest peamine vaade on ettevõtte detailvaade. Tavakasutajatele on nähtavad ainult avalikustatud ettevõtte andmed ehk administraatori poolt kontrollitud ja avalikustatud ettevõtted. Tavakasutaja jaoks on hetkel nähtavad üle 50 ettevõtte.

Ettevõtte detailvaade lihtsustab investori analüüsimist sellega, et rakendus töötleb aastaaruandeid automaatselt ehk investor ei pea enam manuaalselt andmeid ümber trükkima. Lisaks parema ülevaate saamiseks on finantsnäitajad jagatud loogilisteks gruppideks ja numbreid visualiseeritakse graafikutena.

Joonis 1 on näha ettevõtte detailvaate jaotamist erinevateks gruppideks ja Joonis 2 on näha „Capital“ grupi täpsemat vaadet. Erinevad grupid sisaldavad investorite jaoks olulisi finantsnäitajaid ja suhtarve. Joonis 2 on ka näha, et rakenduses on peaaegu igal numbri kohta illustreerivad graafikud. Enne graafiku avamist on näha viimase majandusaasta tulemusi, aga graafikut avades on võimalik suurema perioodi andmeid näha – umbes 10 aasta andmeid.



Joonis 1. Ettevõtte lehe gruppideks jaotamine PG näitel.



Joonis 2. Ettevõtte lehe Capital grupp PG näitel.

Eelnevalt mainituga saab investor väga hea ülevaate, kuidas ettevõttel on ajas läinud ja ettevõttele on võimalik anda esmane hinnang.

Ülejäänud tavakasutaja jaoks nähtavad vaated on portfelli halduseks ning administraatorkasutaja vaated on seotud uue ettevõtte rakendusse lisamisega ja avalikustamisega.

Tuleb tähele panna, et rakendus keskendub ainult USA turule ehk ettevõtete nimekirjas on ainult USA ettevõtted. Lisaks peamiseks fookuseks on ettevõtted, kes maksavad dividende.

Antud rakendus kasutab ära XBRL standardit ja saab aruande andmed masintöötlemisega. Aruanded pärinevad SEC kodulehelt. Rakendus kontrollib uute aruannete olemasolu kord päevas ehk kasutajate jaoks on need kätte saadavad ühe päeva jooksul. Dividendide ja hinna info uuendatakse süsteemis iga kahe tunni tagant.

Rakendusega on võimalik tutvuda järgmisel demo lehel: https://stockbook.xyz/demo.

Antud rakendus on kasutajate jaoks tasuta.

Seoses sellega, et lõputöö autor on ka üks rakenduse autoritest, on töö autor olemasoleva rakenduse ehitamises kaasa löönud. Näiteks lõi tulevaste dividendide vaate, muutis ettevõtte detailvaadet, muutis ettevõtte sisestamise vormi ja lahendas muid väiksemaid ülesandeid.

3.2 Ülevaade rakenduse tehnoloogiatest

Arendatud lahenduses on kasutatud võimalikult palju olemasolevat tehnoloogiat, sest autor jaoks on oluline ühilduvus. Lisaks võib uue tehnoloogia kasutusele võtt koodi segasemaks muuta (näiteks kui kasutada kahte erinevat kasutajaliidese raamistikku korraga) või vajada teenuse täielikku ümberkirjutamist. Ilma mõjuva põhjuseta ei plaani töö autor seda teha.

Rakenduse autorid valisid just all pool nimetatud tehnoloogiad uudishimust ja soovisid neid ka oma rakendusesse arendamisel kasutada, et neid paremini tundma õppida. Eelnevalt olid rakenduse autorid kuulnud valitud tehnoloogiatest ainult head tagasisidet.

Rakendus on jagatud kaheks – kasutajaliideseks ja tagasüsteemiks. Järgmistes alapeatükkides toob töö autor välja olulisemad tehnoloogiad, mida rakenduse ehitusel on kasutatud. Töö autor ei lähe tehnoloogiate kirjeldamisel väga detailseks, sest ei mahu lõputöö skoopi.

3.2.1 Kasutajaliideses kasutatavad tehnoloogiad

Kasutajaliides kasutab Vue.js raamistiku. Vue.js on JavaScript-i raamistik, mille eesmärk on lihtsustada veebirakenduse arendust. Tegemist on paindliku raamistikuga, mida saab jagada erinevateks vaadeteks ja kasutada kasutajaliidese arendamisel erinevates faasides. Näiteks saab Vue.js võtta kasutusele juba olemasolevas kasutajaliideses, ilma et kogu kasutajaliidest peaks ümber kirjutama [4].

Vue.js raamistiku kasutamise eelised on järgmised:

- üks lihtsamaid ja kergemaid (20KB) kasutajaliidese raamistikke;
- integreerub teiste programmeerimiskeeltega väga lihtsalt;
- kerge töötada, kuna sellel on lihtne õppimiskõver;
- kiirem jõudlus tänu raamistiku lihtsusele;
- kasutajaskonna suurenemine;
- hea dokumentatsioon [5].

JavaScript on objektorienteeritud programmeerimiskeel, mida peamiselt kasutatakse kasutajaliideste skriptimiseks ehk tegemiseks.

3.2.2 Tagasüsteemis kasutatavad tehnoloogiad

Tagasüsteem on ülesse ehitatud Go programmeerimiskeeles. Go, tuntud kui golang, tutvustati avalikkusele esimest korda 2009 aastal. Go on mõjutatud C programmeerimiskeelest [6]. Go programmeerimiskeele eelisteks on tema lihtsus. Lisaks eelnimetatud omadustele on Golang populaarsus kasvanud palju – umbes 147% (2018-2019 aastal) [7].

Andmete salvestamiseks kasutab tagasüsteem Amazon DynamoDB. DynamoDB on NoSQL ehk mitterelatsiooniline võti-väärtus andmebaas. DynamoDB eeliseks on

andmebaasi skeemi puudumine, mis on eriti kasulik arenduse alguses, kui pole struktuuri teada. Lisaks DynamoDB andmebaas on täielikult hallatud Amazoni poolt [8].

3.3 Rakenduse konkurendid

Turul on palju tasuta ja tasulisi rakendusi, mis on arendatud rakendusele konkurendiks. Mõned näited tasuta finantsanalüüsi rakendustest on järgmised:

- TradingView;
- TD Ameritrade;
- StockCharts.com;
- Yahoo Finance;
- Google Finance;
- FINVIZ [9].

Investorite jaoks on igal konkureerival rakendusel on omad eelised ja puudused. Investorite jaoks on osades rakendustes liiga vähe ja teistes liiga palju infot ettevõtete kohta ning rakendused lahendavad erinevaid probleeme investorite jaoks. Ehk investorite eelistused on väga erinevad. Muidu on rakenduste idee sarnane: näidata olulisi finantsandmeid ja suhtarve ning võivad pakkuda lisa funktsionaalsusi.

Autor usub, et nii antud rakenduse kui ka konkurentide probleemiks on olnud finantsandmete kaardistamine. Seda, kuidas keegi seda probleemi lahendas on raske teada, sest see pole avalik info. Sellepärast pole autoril ka võimalik konkurentide lahendusega tutvuda ja neid võrrelda ning vastavalt antud rakenduse administraatorite vajadustele tuli leida oma lahendus.

4 Finantsandmete automatiseeritud kaardistamise lahendus

Lahenduse arendamisel kasutati agiilset ja testipõhist arendus metoodikat. Agiilsel tarkavaraarenduse põhimõtetest jälgiti järgmisi põhimõtteid:

- tarkvara tarnitakse võimalikult tihti ja kiiresti;
- lahenduse osad on jagatud eraldiseisvateks osadeks, et hiljem on võimalik laiendada;
- lihtsus – ebavajaliku töö mitte tegemine;
- muutuvate oludega arvestamine [10].

Tagasüsteemi lisatud loogika on lahendatud testipõhise arendus metoodika järgi ehk esimesena kirjutatakse testid ja teisena lisatakse loogika ning parandatakse testid ja kolmandaks refaktoreeritakse kood [11]. Lisatud ühiktestidega kontrollitakse 90% töö autori poolt lisatud funktsionaalsusest tagasüsteemis.

Töö autor on jaganud finantsandmete automatiseeritud kaardistamise lahenduse järgmisteks osadeks:

- valemite kaardistamine;
- kõikide finantsandmete kuvamine;
- kasutatud valemite kuvamine;
- valemite muutmine;

Järgnevad muudatused on kättesaadavad ja nähtavad ainult administraatorkasutajale. Töö autori järgnevad muudatused on seotud kasutajaliidese ja tagasüsteemiga.

4.1 Valemite kaardistamine

Valemi kaardistamise eesmärk on koodis oleva arvutamise loogika viimine andmebaasi nii, et iga valemi muudatuse peale ei peaks arendaja koodi muutma.

Valemite kaardistamiseks tuli töö autoril lahendada järgmised ülesanded:

- valemite grupeerimine;
- loogika valemite arvutamiseks;
- valemite kasutamine;
- valemite testimine.

Järgnevad muudatused rakenduses on seotud ainult tagasüsteemiga.

4.1.1 Valemite grupeerimine

Esimese asjana tutvus töö autor koodiga, et mõista mitmeks grupiks on mõistlik valemid jagada. Töö käigus tuli välja, et kaardistamise valemid tuleb jagada kaheks grupiks – vaikimisi ja ettevõtte põhiseks. Ettevõtete põhised valemid on seotud kindla ettevõttega. Juhul, kui kindlal ettevõttel on ettevõtte põhine valem, siis kasutatakse seda valemit. Vastasel korral kasutatakse vaikimisi valemit, et finantsandmeid arvutada.

Töö autoril tekkis ka idee jagada ettevõtte põhised valemid omakorda aasta põhiseks. Hiljem selgus, et see ei ole mõistlik, sest eelnevalt pole olnud vajadust aasta põhisteks valemiteks ja ilma vajaduseta ei ole mõistlik lisa keerukust lisada. Nii jäi töö autor kindlaks, et valemid tuleb jagada kaheks.

Alguses proovis töö autor vaikimisi valemid üldisemaks teha. Näiteks kui väli sisaldab mingit kindlat sõna, siis need väljad arvutatakse kokku. Kahjuks oli sarnaste nimedega välju palju, mis olid mõnel juhul teise välja täpsustuseks või lihtsalt sama väli lihtsalt teise nimega. Nii tuli vaikimisi valemid kokku panna kindlate välja nimedega.

Järgmise sammuna töö autor kogus finantsandmete arvutamise loogika eelmises lõigus loodud gruppideks. Sellega sai töö autor hea ülevaate kõikidest valemi tingimustest. Valemi ühtse struktuuri paika panemiseks kaardistas töö autor kõik valemi arvutamise tehted. Finantsandmete arvutamises kasutati järgmisi tehteid:

- aritmeetilised tehted (+, -, /, *);
- välja väärtuse kontroll;
- kahe välja väärtuse võrdlus (<, > ja =);

- maksimaalse ja minimaalse välja väärtuse saamine;
- välja väärtuse positiivseks tegemine;
- eelnevalt arvutatud välja väärtuse kasutamine uues valemis.

Teades kõikvõimalikke tehteid oli töö autoril võimalik luua ühtse struktuuriga valemeid. Tabel 2 on välja toodud töö autori poolt loodud ühtse struktuuriga võimalikud valemi tehted koos selgitustega ja näidetega.

Ühtse valemi struktuuri loomisel jälgis töö autor järgmist reeglit: algab matemaatilise funktsiooniga, mis on ümbritsetud sulgudega. Töö autor valis just sellise struktuuri , sest sama stiil on juba kasutusel ka muudes rakendustes (näiteks Excelis). Nii said ka teised matemaatilised funktsioonid samale stiilile viidud. Tuleb tähele panna, et eelnev reegel ei kehtinud aritmeetilistele tehetele, sest autor soovib nende struktuuri jätta samaks.

Tabel 2. Võimalikud valemi tehted koos selgituste ja näidetega.

Tehe	**Selgitus**	**Näide**
OR	Sisendiks nimekiri välja nimedest. Tagastab esimese välja, mis ei võrdu nulliga.	OR(väli1, väli2)
AND	Vajab kahte sisendit. Esimeseks sisendiks on nimekiri välju, mis on ümbritsetud nurksulgudega. Juhul kui nimekirjas ühe välja väärtus on null, siis tagastatakse null. Vastasel korral tagastatakse teise sisendi väärtus.	AND([väli1, väli2], väli3)
MIN	Sisendiks nimekiri välju. Tagastab kõige väikesema väärtusega välja.	MIN(väli1, väli2)
MAX	Sisendiks nimekiri välju. Tagastab kõige suurema väärtusega välja.	MAX (väli1, väli2)
COMPARE	Vajab kolme sisendit. Kaheks esimeseks sisendiks on väljad, mida võrreldakse. Kolmandaks sisendiks on võrdlusmärk. Lubatud võrdlusmärkideks on <, > ja =.	COMPARE(väli1, väli2, <)
ABS	Sisendiks väli, mille väärtust soovitakse positiivseks teha.	ABS(väli1)
+, -, /, *	Tavalised aritmeetilised tehted.	(väli1 – väli2) / väli3
$	Selleks, et eelnevalt defineeritud valemit teises valemis kasutada, tuleb see ümbritseda dollari märgiga.	Väli1 + $defineeritudVäli$

Kõik Tabel 2 nimetatud valemi tehteid on võimalik ka omavahel kombineerida. Joonis 3 on näha näidet kombineeritud valemist.

$$väli1 + MAX(väli2, OR(väli3, väli4))$$

Joonis 3. Näide kombineeritud valemist.

Lõpuks sai töö autor paika valemi grupid ja struktuuri, mille tulemusel sai valemid kirjutada vastavalt loodud grupile ja struktuurile. Valemeid kokku pannes märkas autor, et mõni andmeväli vaja tagavara valemeid. See on vajalik, sest igal ettevõttel ei pruugi ühesuguseid välju olla. Sellepärast, et süsteem oskaks valida esimese valemi, mille tulemus ei ole null. Selleks pidi töö autor ühe valemi asemel looma valemite nimekirja.

Joonis 4 on näha näidet loodud ettevõtte põhisest valemist, mis on andmebaasi salvestatud. Näites on kasutatud uut valemite struktuuri, valemite kombineerimist ja tagavara valemite lisamise võimalust.

```
{
        "ShortTermInvestments": {
            "formulas": [
                "AvailableForSaleSecurities",
                "ShortTermInvestments",
                "OR(MarketableSecuritiesCurrent, MarketableSecurities) + OtherShortTermInvestments",
                "AvailableForSaleSecuritiesCurrent + TradingSecuritiesCurrent + AvailableForSaleSecuritiesDebtSecuritiesCurrent"
            ],
            "context_keys": [
                "AvailableForSaleSecurities",
                "ShortTermInvestments",
                "MarketableSecuritiesCurrent",
                "MarketableSecurities",
                "OtherShortTermInvestments",
                "AvailableForSaleSecuritiesCurrent",
                "TradingSecuritiesCurrent",
                "AvailableForSaleSecuritiesDebtSecuritiesCurrent"
            ],
            "round": -1
        }
}
```

Joonis 4. Ettevõtte põhine valem PG näitel.

Andmebaasi salvestamisel salvestati ka „context_keys“ ja „round“ väljad, mis on hiljem vajalikud valemi arvutamiseks. Väli „context_keys“ sisaldab kõiki unikaalseid väljade nimesid valemis ja „round“ näitab mitu numbrit peale koma kohta ümardatakse. Juhul, kui ei soovita ümardada, siis märgistatakse selle väärtus miinus üheks.

Kokku tuli üks vaikimisi ja kaheksa ettevõtte põhist valemite dokumenti Üks dokument võis sisaldada kuni 18 valemit.

Pärast valemite ühtsesse struktuuri paika panemist tuli valemid andmebaasi salvestada, mis teostati arenduse käigus. Ettevõtte põhised valemid salvestati ettevõtete kaupa, sest

töö autor soovis ettevõtteid eraldi hoida. Lisaks andmebaasi dokumendid, mis sisaldavad mitme ettevõtte põhiseid valemeid, võivad aja jooksul mahult kasvada.

4.1.2 Valemitega arvutamine

Peale ühtse valemi struktuuri paika panemist ja nende salvestamist andmebaasi oli vaja see ka süsteemile arusaadavaks teha. Selleks hakkas töö autor otsima programmipakette, mis lahendaks arvutamise loogika. Programmipaketti leidmisega polnud vaja „ratast leiutada“ vaid sai kasutada olemasolevat programmi koodi.

Otsingu tulemusel leidis töö autor ühe programmipaketti - Goexpression [12]. Leitud programmipakett lahendas muutujatega aritmeetilise tehetega arvutamise. Antud paketiga on võimalik väärtusi võrrelda, aga see ei sobinud anutud rakenduse jaoks, sest Goexpression-it kasutades pole võimalik muutujaid kaasa anda ja see ei tagastanud numbrilist väärtust vaid tagastatakse puu kujulise struktuurina, mida peab edasi arvutama.

Töö autor proovis leida ka matemaatilistele funktsioonidele programmipaketti. Kahjuks ei leidnud ühtegi, mis sobiks antud probleemi lahendamiseks. Nii tuli autoril see loogika ise kirjutada.

Leitud programmipaketti sai kaasa anda valemid, mis sisaldavad aritmeetilisi tehteid koos objektiga, mis sisaldab muutujate väärtusi. Selleks oli eelnevalt oluline salvestada „context_keys“ valemi dokumenti, et hiljem ei peaks unikaalseid muutujaid otsima ja oleks lihtsam neile väärtusi määrata. Finantsandmete ühtsema ümardamise hoidmiseks, lisas autor ka „round“ välja, sest see oli eelnevalt segaselt koodis laiali kirjutatud.

Selleks, et leitud programmipaketti kasutada, on vaja eelnevat töö autori loodud matemaatilised funktsioonid aritmeetilisteks valemiteks konverteerida. Matemaatiliste funktsioonide mõistmiseks oli autoril kaks valikut. Kas täht haaval läbi käimine või regulaaravaldiste kasutamine. Täht haaval läbi käimine muudab koodi loetavust halvemaks võrreldes regulaaravaldiste kasutamine. Lisaks on regulaaravaldis loodud teksti mustrite tuvastamiseks. Seoses sellega, et loodud valemid olid kindla mustriga, otsustas autor regulaaravaldiste kasuks.

Töö autor pidi matemaatiliste funktsioonide loogika eraldi defineerima, sest kõigil neil oli oma käitumine. Kood ehitati nii ülesse, et tulevikus uute matemaatiliste funktsioonide

lisamine oleks võimalikult lihtne. Joonis 5 on näha, et tuleb lihtsalt lisada uus tehtemärk ja implementeerida matemaatilise funktsiooni arvutamise loogika.

```
func calculateBasedOnOperator(match *regexp2.Match, contextKeys map[string]interface{}) string {
    matchFormula := addBracesAtEnd(match.GroupByNumber(3).String())
    switch operator := strings.ToUpper(match.GroupByNumber(1).String()); operator {
    case OperatorAnd:
        return getAndOperatorValue(matchFormula, contextKeys)
    case OperatorOr:
        return getOrOperatorValue(matchFormula, contextKeys)
    case OperatorCompare:
        return getCompareOperatorValue(matchFormula, contextKeys)
    case OperatorAbs:
        return getAbsOperatorValue(matchFormula, contextKeys)
    case OperatorMax:
        return getMaxOperatorValue(matchFormula, contextKeys)
    case OperatorMin:
        return getMinOperatorValue(matchFormula, contextKeys)
    default:
        return ""
    }
}
```

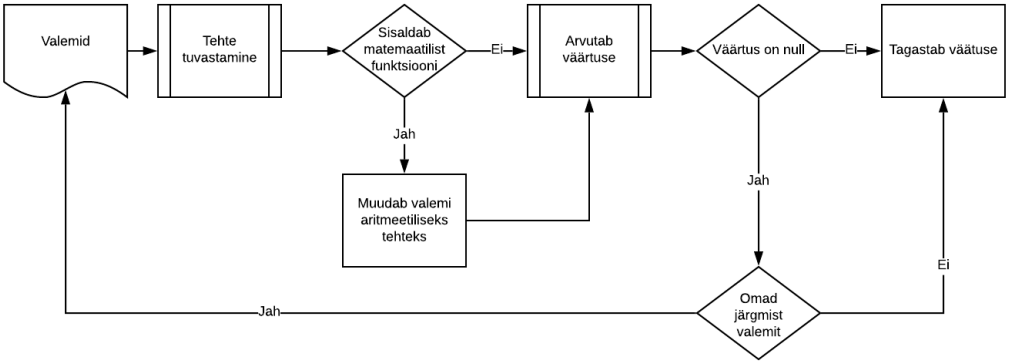
Joonis 5. Funktsioon tehtemärkidega arvutamiseks.

Töö autori poolt loodud valemite arvutamine käib järgmistes etappides:

1. tuleb sisendiks nimekiri välja valemitest;
2. valitakse esimene valem;
3. tuvastatakse regulaaravaldisega, kas valem sisaldab töö autori loodud matemaatilisi funktsioone;
4. jah – liigub etappi nr 5 , ei - liigub etappi nr 6;
5. muudad matemaatilised funktsioonid aritmeetilisteks teheteks;
6. arvutab väärtuse – kasutades leitud programmipaketti;
7. tuvasta kas tulemuse väärtus on null;
8. jah – liigub etappi nr 9, ei – liigub etappi nr 11;
9. kontrollib tagavara valemi olemasolu;
10. jah – valitakse järgmine valem nimekirjast ja liigub etappi nr 3, ei – liigu etappi nr 11;

11. tagastab arvutatud väärtuse;

Antud kirjeldatud arvutamise skeem on kirjeldatud Joonis 6.

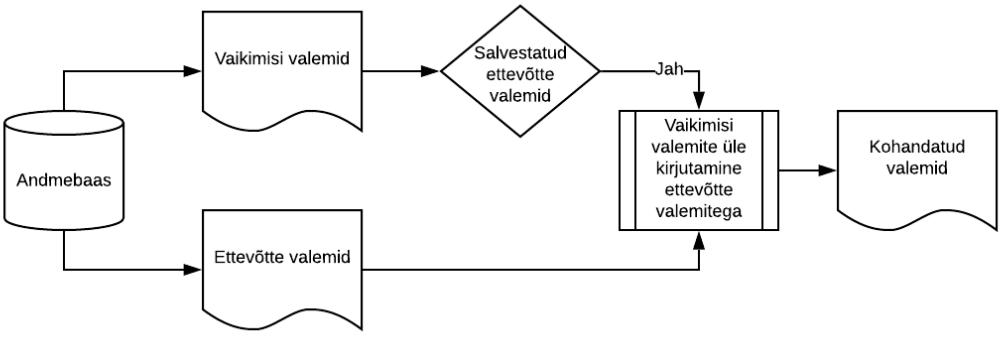


Joonis 6.Valemite arvutamise vooskeem.

4.1.3 Valemite kasutamine

Selles osas pidi töö autor kahe eelmise alapeatüki lahendused ühendama.

Esiteks tuli teha loogika, mis küsib andmebaasist vaikimisi ja ettevõtte põhist valemit. Enne valemite kasutamist on oluline valemid vastavalt ettevõtte valemitele kohandada. Päritud vaikimisi valemid kirjutatakse ettevõtte põhiste väljade valemitega üle ja nii saadakse kohandatud valemid. Juhul, kui ettevõtte põhiseid valemid ei ole, siis ei kirjutata vaikimisi valemist midagi üle. Joonis 7 on näha skeemi kohandatud valemi koostamise kohta.



Joonis 7. Kohandatud valemi loomine.

Järgmiseks oli vaja olemasoleva valemite arvutamise loogika asendada kohandatud valemitega ja ühendada valemi arvutamise loogikaga. Selle tulemusena sai töö autor

koodis palju vana koodi eemaldada. Vana koodi eelmaldamine muutis koodi loetavuse paremaks, sest ühte välja arvutatakse ainult ühes kohas – enne oli arvutatud ühte välja mitmes kohas. Joonis 8 on näha osa vanast koodist, kus ühe finantsandme kaardistamiseks on loodud mitu koodi rida. Joonis 9 on näha töö autori poolt muudetud koodi. Võrreldes neid omavahel on näha, et loetavus on paranenud ja koodiridade arv on märgatavalt vähenenud.

```
func (fy *FiscalYear) MapToFiscalYear(m xbrl.FactsMap) {
    fy.mapDeprecationAndAmortization("DepreciationAndAmortizationAndImpairmentOfGoodwillAndIntangibleAssets", m)
    fy.mapDeprecationAndAmortization("DepreciationAmortizationAndOther", m)
    fy.mapDeprecationAndAmortization("DepreciationAmortizationAndAccretionNet", m)
    fy.mapDeprecationAndAmortization("DepreciationDepletionAndAmortization", m)
    fy.mapDeprecationAndAmortization("DepreciationAndAmortization", m)
    fy.calculateDeprecationAndAmortization(m)
    fy.mapInterestExpense("InterestAndDebtExpense", m)
    fy.mapInterestExpense("InterestExpense", m)
    fy.mapInterestExpense("InterestExpenseDebtExcludingAmortization", m)
    fy.mapInterestExpense("InterestAndOtherDebtExpenseIncomeNet", m)
    fy.mapInterestExpense("InterestExpenseDebt", m)
    fy.mapInterestExpense("InterestIncomeExpenseNet", m)
    fy.calculateInterestExpense2(m)
    fy.mapOperatingIncome("OperatingIncomeLoss", m)
    fy.calculateOperatingIncome(m)
    fy.addInterestAmortization(m)
    fy.mapNetIncomeToFiscalYear("NetIncomeLossAvailableToCommonStockholdersBasic", m)
    fy.mapNetIncomeToFiscalYear("NetIncomeLoss", m)
    fy.mapNetIncomeToFiscalYear("ProfitLoss", m)
    fy.mapNetIncomeToFiscalYear("ComprehensiveIncomeNetOfTaxIncludingPortionAttributableToNoncontrollingInterest", m)
```

Joonis 8. Vana kood finantsandmete kaardistamiseks.

```
func (fy *FiscalYear) MapToFiscalYear(m xbrl.FactsMap, customMap formula.Map) UsedFormulaAggregate {
    usedFormulas := make(formula.UsedMap, 100)
    fy.DepreciationAndAmortization = usedFormulas.GetFormulaResult("DepreciationAndAmortization", customMap, m)
    fy.InterestExpense = usedFormulas.GetFormulaResult("InterestExpense", customMap, m)
    fy.OperatingIncome = usedFormulas.GetFormulaResult("OperatingIncome", customMap, m)
    fy.NetIncome = usedFormulas.GetFormulaResult("NetIncome", customMap, m)
    fy.Revenue = usedFormulas.GetFormulaResult("Revenue", customMap, m)
    fy.ShareHoldersEquity = usedFormulas.GetFormulaResult("ShareHoldersEquity", customMap, m)
    fy.ShareHoldersEquityMinorityInterest = usedFormulas.GetFormulaResult("ShareHoldersEquityMinorityInterest", customMap, m)
    fy.SharesOutstandingDiluted = usedFormulas.GetFormulaResult("SharesOutstandingDiluted", customMap, m)
    fy.CashAndCashEquivalents = usedFormulas.GetFormulaResult("CashAndCashEquivalents", customMap, m)
    fy.ShortTermInvestments = usedFormulas.GetFormulaResult("ShortTermInvestments", customMap, m)
    fy.DebtLongTerm = usedFormulas.GetFormulaResult("DebtLongTerm", customMap, m)
    fy.DebtShortTerm = usedFormulas.GetFormulaResult("DebtShortTerm", customMap, m)
    fy.CashProvidedByOperatingActivities = usedFormulas.GetFormulaResult("CashProvidedByOperatingActivities", customMap, m)
    fy.CapitalInvestments = usedFormulas.GetFormulaResult("CapitalInvestments", customMap, m)
    fy.EarningsBeforeIncomeTaxes = usedFormulas.GetFormulaResult("EarningsBeforeIncomeTaxes", customMap, m)
    fy.EarningsPerShare = usedFormulas.GetFormulaResult("EarningsPerShare", customMap, m)
    fy.TaxRate = usedFormulas.GetFormulaResult("TaxRate", customMap, m)
    fy.DividendPerShare = usedFormulas.GetFormulaResult("DividendPerShare", customMap, m)

    return UsedFormulaAggregate{
        FormulaMap: usedFormulas,
        Year:       fy.FiscalYear,
    }
}
```

Joonis 9. Uus kood finantsandmete kaardistamiseks.

4.1.4 Valemite testimine

Arendatud koodi testimine on rakenduse autorite, kui ka töö autori jaoks väga tähtis.

Esmast kontrollis töö autor, et ei lõhkunud olemas olevaid teste ja et nende tulemused oleksid samad.

Rakenduse autorid olid varasemalt kirjutanud testid finantsandmete arvutamiseks. See oli töö autori jaoks väga hea, sest sellega sai kohe testida, et asendatud valemid ei muutnud finantsandmete tulemusi. Siinkohal oli töö autoril oluline ka salvestada ümardamise täpsus andmebaasi, sest muidu oleks testides palju ümardamise vigu.

Lisaks lisas töö autor uusi teste, et valideerida kõiki tehteid ühekaupa. Sellega tagab töö autor, et üheski tehtes poleks arvutamise vigu. Testimine andis ka lisa kindlust töö autorile, et valemite arvutamise loogika töötab nii nagu on vaja. Kokku testiti 18 erinevat valemi kombinatsiooni.

4.2 Kõikide finantsandmete kuvamine

Aruannete finantsandmeid on oluline rakenduses kuvada, et administraatoril oleks võimalik valemis viga üles leida ja hiljem see ka ära parandada.

Finantsandmete kuvamiseks tuli töö autoril lahendada järgmised ülesanded:

- ettevõtte kõikide finantsandmete salvestamine andmebaasi;
- kõikide finantsandmete kättesaadavaks tegemine;
- vaade kasutajaliidesesse.

Nagu ülesannetes on näha, siis töö autor pidi tegema muudatusi nii kasutajaliideses kui ka tagasüsteemis.

4.2.1 Kõikide finantsandmete salvestamine andmebaasi

Töö autor otsustas ettevõtte finantsandmed salvestada andmebaasi, et kasutajaliidese päringud oleksid kiiremad. Andmebaasi päring on rakenduse jaoks optimaalsem ja kiirem, kui iga kord aruanded alla laadida ja töödelda.

Nii otsustas töö autor, et kõige õigem aeg on salvestada finantsandmed andmebaasi järgmistel olukordadel:

- uue ettevõtte sisestamisel;

- ettevõtte uue majandusaasta aruannet importides;
- ettevõtte finantsandmete üle arvutamisel;

Need on hetked, kus ettevõtte aruanded laetakse alla ja töödeldakse. Sel hetkel on kõik vajaminevad finantsandmed olemas ja hea võimalus andmebaasi salvestamiseks. Nende hetkete kasutamisega on võimalik vältida lisatöö tegemisest ja lihtne võimalus andmeid aktuaalsena hoida – uute ja parandusaruannete korral uuendatakse ka kõik finantsandmeid andmebaasis.

Enne aruannete salvestamist pidi töö autor aruandeid töötlema, et sinna ei salvestata midagi muud kui numbrilisi finantsandmeid. Oluline oli töödelda aruande finantsandmeid, et administraator ei peaks ebaolulisi välju nägema ja vähendada andmebaasi salvestatud andmete mahtu. Hilisem töötlus oleks iga päringu juure olnud lisatöö, mis ei oleks olnud kõige mõistlikum.

Salvestamise käigus tuli välja, et ettevõtte üks aastaaruanne võib sisaldada suurt hulka infot (olenevalt ettevõttest paarist sajast kuni poole tuhandeni andme väljani). Sellest tulenevalt otsustas töö autor, et ettevõtte kõik aruanded tuleb aastate põhiselt salvestada.

Ühe kaupa aasta põhiste dokumentide salvestamine ei tundunud töö autori arvetes mõistlik ja soovis seda protsessi veidi optimeerida. Selleks kirjutas töö autor funktsiooni, millega on võimalik mitu dokumenti korraga andmebaasi salvestada. Piiranguks seadis korraga salvestada 25 dokumenti, mis on ka Amazon DynamoDB limiit [13]. Juhul, kui saadetakse funktsiooni rohkem dokumente, siis töö autor tegi nii, et dokumendid salvestatakse ikkagi 25 kaupa ja tehakse mitu erinevat päringut.

Funktsiooni, mis võimaldab korraga mitu dokumenti salvestada, saab ka järgmistes lahendustes kasutada.

4.2.2 Kõikide finantsandmete kättesaadavaks tegemine

Oluline on kõik finantsandmed administraatorile kättesaadavaks teha, et tal oleks võimalik näha väljade nimesid ja väärtusi. Seda infot kasutab administraator finantsandmete kaardistamise valemeid koostades.

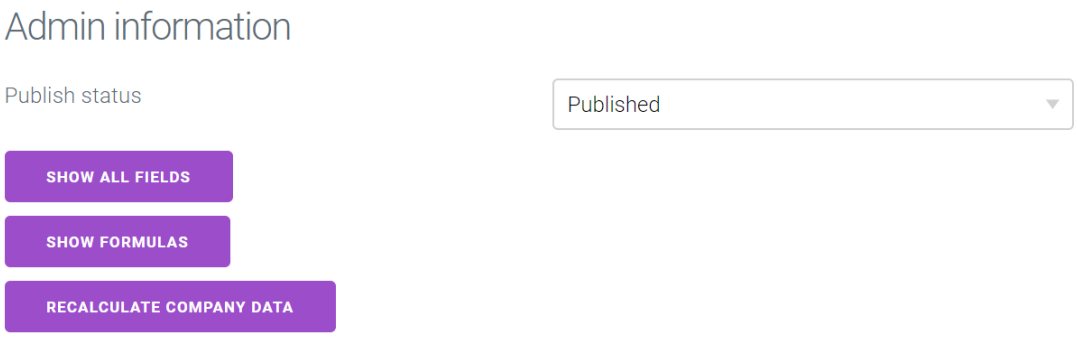
Selleks tuleb eelmises alapeatükis salvestatud andmed andmebaasist välja pärida. Pärimise aluseks on ettevõtte börsitähis ja periood.

Töö autor tegi kõik finantsandmed kätte saadavaks läbi API endpointi, mis on kaitstud rakenduse administraatori õigusega, sest see vaade on mõeldus ainult neile.

4.2.3 Vaade kasutajaliidesesse

Nagu töö autor juba eelnevalt mainis, et vaade on administraatori töö lihtsustamiseks – finantsandmete kaardistamise valemite vea otsimiseks ja parandamiseks. Seoses sellega on ka kasutajaliideses vaade nähtav ainult administraatorkasutajatele.

Põhjusel, et vaade on seotud ettevõttega, siis töö autor tegi otsetee vaatele ettevõtte lehele "Admin information" sektsiooni – Joonis 10 . See tundus töö autori jaoks kõige mõistlikum, sest on seotud ainult administraatorkasutajaga ja on kergesti kätte saadav.



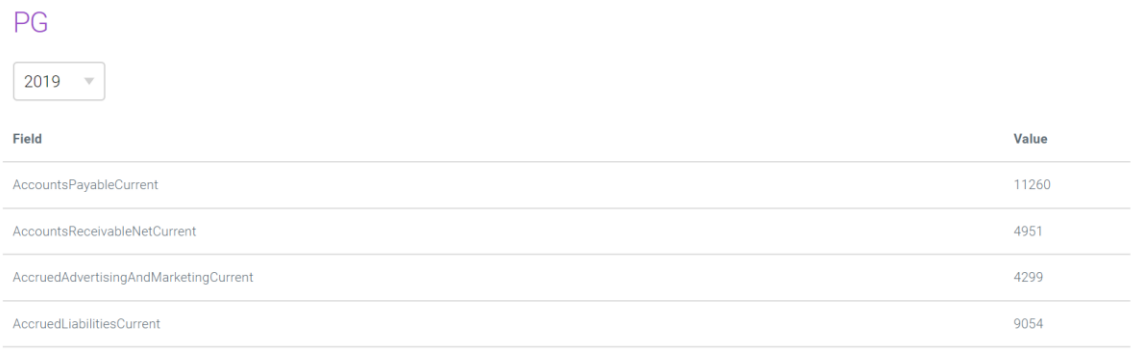
Joonis 10. Otsetee kõikidele finantsandmetele ja kasutatud valemitele.

Selleks, et administraatorkasutajal mugavam vaadet kasutada oleks, lisas töö autor paar mugandavat funktsionaalsust:

- vaade avaneb alati uues aknas, sest finantsandmeid võrreldakse alati mingi muu näitaja vastu;
- vaate üleval on näha ettevõtte börsitähist, kui mitu akent lahti, siis teada, mis ettevõtte andmetega on tegu;
- eelmises punktis mainitud ettevõtte börsitähise tegemine otseteeks – võimalik lihtsalt ettevõtte lehele tagasi minna;
- vaate avades näitab automaatselt viimase aasta aruande andmeid;
- aastate valikuriba on ettevõtte põhine – olenevalt, mitu majandus aastaaruannet rakenduses on.

Selleks, et iga ettevõttel oleks personaalne aastate valik pidi autor tegema eraldi endpoindi, et neid välja pärida. Seda päriti andmebaasis olemasolevate majandusaasta aruannete pealt. Seda funktsionaalsust on võimalik ka tulevikus kasutada.

Kasutajaliidese vaade kirjutati eraldi vaatesse ehk on eraldi seisev teistest funktsionaalsusest. Selleks, et lisatud vaade sobiks kokku eelnevate vaadetega kasutas töö autor vaate loomiselt rakenduse stiile. Lõpptulemust on näha Joonis 11.



Joonis 11. Kõikide finantsandmete vaade kasutajaliideses PG näitel.

4.3 Kasutatud valemite kuvamine

Antud osa on oluline administraatorkasutaja jaoks, et neil oleks lihtsam aru saada, kuidas mingi finantsnäitaja arvutati.

Kasutatud valemite kuvamiseks tuli töö autoril lahendada järgmised ülesanded:

- kasutatud valemite salvestamine andmebaasi;
- kasutatud valemite kättesaadavaks tegemine;
- vaade kasutajaliidesesse.

Järgnevad muudatused on seotud nii kasutajaliidese ja tagasüsteemiga.

4.3.1 Kasutatud valemite salvestamine

Finantsnäitaja võib koosneda ühest või mitmest tagavara valemist ning valemid võivad sisaldada mitmeid kombineeritud matemaatilisi tehteid ja funktsioone. Finantsandmete kontrollimisel on oluline, et administraator näeb täpselt, kuidas välja väärtus kujunes. Muidugi on administraatoril võimalik valemite põhjal arvutada, milline tagavara valem

valiti ja millise välja väärtusi arvutamisel kasutati. Näiteks Joonis 3 pole teada, mis väärtus tagastatakse OR või MAX funktsioonis. Selleks peaks administraator otsima välja kõik välja väärtused ja vaatama, millise välja väärtuse lõpuks valiti. Selline valemite läbi arvutamine on administraatori jaoks lisa ajakulu, mida töö autor ei tahtnud tekitada. Nii otsustas töö autor salvestada finantsnäitaja arvutamisel kasutatud valemid andmebaasi, et hiljem neid kasutajaliideses kuvada.

Kasutatud valemite andmebaasi salvestamise põhjus ja olukord on sama, mis oli kirjeldatud "4.2.1 Kõikide finantsandmete salvestamine andmebaasi" alapeatükis.

Mainitud peatükis loodi ka võimalus salvestada mitu dokumenti korraga. Töö autor kasutas antud funktsionaalsuse ära ja salvestas kasutatud valemid aastate kaupa andmebaasi.

4.3.2 Kasutatud valemite kättesaadavaks tegemine

Kasutatud valemid on oluline kättesaadavaks teha, et administraator näeks, mis kaardistamise valemit kasutati finantsnäitaja arvutamisel. See lihtsustab administraatoril finantsnäitajate kaardistamise kontrollimist.

Antud alapeatükk on samuti sarnane "4.2.2 Kõikide finantsandmete kättesaadavaks tegemine" alapeatükkiga. Ainuke erinevus on endpointi URL-s ja tagastatavates andmetes.

4.3.3 Vaade kasutajaliidesesse

Kasutatud valemid on seotud kindla ettevõttega. Otsetee kasutatud valemitele vaatele lisati ettevõtte lehel „Admin information" sektorisse – Joonis 10. Antud sektor on nähtav ainult administraatoritele.

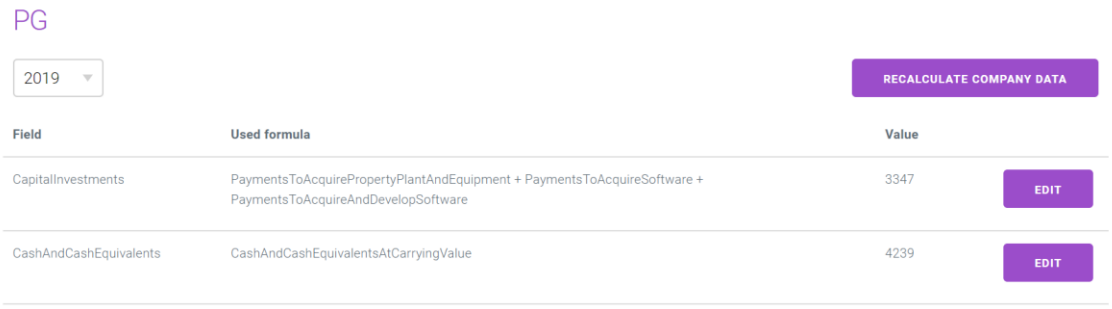
Kasutatud valemid on kõikides eelnevast loogikast eraldatud eraldi vaatesse. Selles vaates on võimalik näha järgmisi andmeid:

- välja nimi, mille kohta valem käib;
- millist valemit lõpuks arvutamisel kasutati;
- kokku arvutatud väärtus;

- otsetee kindla välja valemi muutmiseks – täpsemalt järgmises alapeatükis.

Ülejäänud vaate mugandused ja stiil on sama eelnevalt loodud vaatega.

Töö autori lahendust kasutajaliideses on võimalik näha Joonis 12.



Joonis 12. Kasutatud valemite vaade kasutajaliideses PG näitel.

4.4 Valemite muutmine

Finantsnäitajad pole alati õigesti kaardistatud ja selle parandamiseks on vaja muuta kaardistamist. Selleks, et administraatorkasutajal oleks võimalik valemeid muuta on vaja kõikide eelnevate lahenduste olemasolu järgmistel põhjustel:

- valemite kaardistamine – vaja teada loodud valemite struktuuri ja süsteem peab oskama seda tõlgendada;
- kõikide finantsandmete kuvamine – näha välju ja välja väärtusi, mida valemis saab kasutada;
- kasutatud valemite kuvamine – näha, mis valemit kasutati ja vajadusel valemeid muuta.

Valemite muutmise võimaldamiseks tuli töö autoril lahendada järgmised ülesanded:

- valemite pärimine ja salvestamine;
- vaade kasutajaliidesesse;
- valemi muutmise testid.

4.4.1 Valemite pärimine ja salvestamine

Valemite pärimine ja salvestamine on taaskord kaitstud rakenduse administraatorkasutaja õigustega.

Seoses sellega, et valemid pole aastate põhised, siis valemi pärimisel saadetakse kogu dokument kõikide väljade valemitega.

Ettevõtte valemite pärimisel tuleb tähele panna, et olukorras, kus ettevõtte põhist valemit ei ole antakse automaatselt vaikimisi valemid. Juhul kui on, siis kirjutatakse vaikimisi valemid ettevõtte põhistega üle ehk luuakse kohandatud valemi dokument. See kõik on administraatorkasutaja jaoks märkamatu.

Autor lahendas valemite muutmise hetkel nii, et on võimalik muuta ainult ettevõtte põhiseid valemeid. Vaikimisi valemeid muuta on hetkel liiga ohtlik, sest valemi muutja ei tea, kuidas võib ühe valemi muutmine teiste ettevõtete finantsandmete kaardistust muuta.

Peale valemite kaardistamise kasutusele võttu tuleb rakenduse administraatorkasutajatel jälgida, kas tekib ka vajadus vaikimisi valemite muutmiseks. Juhul kui tekib, siis on see hilisem arendus, mis lõputööd ei sisalda.

Isegi, kui ennem ei olnud ettevõtte põhist valemit ja pärimisel anti vaikimisi valem, siis salvestamise korral salvestatakse automaatselt ettevõtte põhiseks valemiks.

Salvestamisel tuleb tähele panna, et muudetud valemi välja väärtus ei muutu automaatselt, sest peale kõikide valemite muudatust tuleb andmed uuesti üle arvutada. Töö autor ei soovinud seda automaatseks teha, sest iga muudatuse/salvestamise peale 10+ aastat uuesti arvutada on liiga kulukas.

4.4.2 Vaade kasutajaliidesese

Ettevõtte põhiseid valemid on võimalik muuta kasutatud valemite vaatest. Kus iga välja lõpus on „EDIT“ nupp – Joonis 12. See on otsetee kindla välja valemi muutmiseks.

Vastavalt saadud välja valemile on administraatorkasutajal võimalik valemeid muuta, eemaldada või lisada.

Töö autor otsustas valemi kaardistamise muutmise teksti väljana lahendada järgmistel põhjustel:

- kõikide finantsandmete väljade salvestamisel tuli välja, et välju on palju (100 + välja) ja selle otsimine ja valimine vaates on administraatori jaoks tülikas ehk kõiki välju ei ole mõttekas näidata valemite kaardistamise vaates;
- kõikide väljade näitamisel oleks oluline ka näidata iga välja väärtust, aga selleks vaatesse piisavalt ruumi polnud;
- matemaatiliste tehete ja funktsioonide valimine nupuga ei tundunud antud juhul mõistlik, sest tehted ja funktsioonid on lihtsa struktuuriga;
- valem on salvestatud teksti kujul, siis on mõttekas seda ka näidata ja muuta teksti kujul.

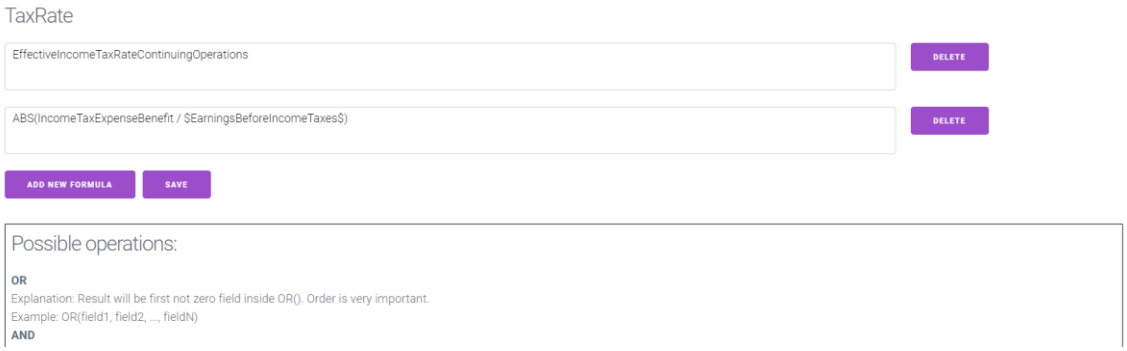
Valemi salvestamise korral kontrollitakse kas valem vastab järgmistele tingimustele:

- valem ei tohi olla sama – siis ei salvestata midagi;
- peab omama vähemalt ühte valemit ehk ei pea olema tagavara valemeid;
- valem või tagavaravalem ei tohi olla tühi;
- sulud oleksid suletud;
- kõik valemid (isegi tagavara) peavad olema arvutatavad – süsteem arvutab tagataustal testandmetega läbi.

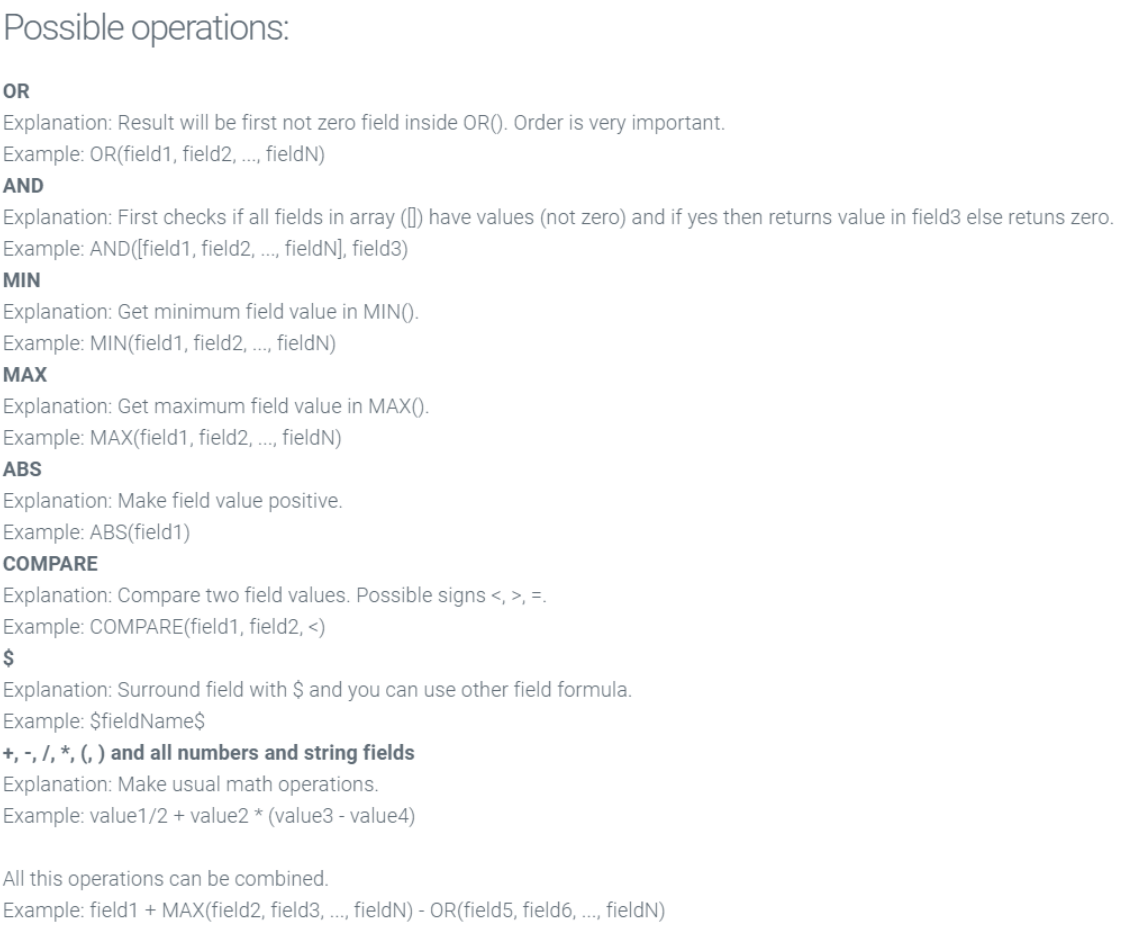
Vastavalt valemi kontrollile näidatakse salvestamise tulemus. Sellega saab valemi muutja teada, kas salvestamine õnnestus või ei. Ebakorrektset valemit pole võimalik salvestada.

Lisaks sellesse vaatesse on lisatud ka tehete selgitused, juhul kui administraatoril peaks meelest ära minema - Joonis 14.

Joonis 13 on võimalik näha töö autori loodud valemi muutmise vaadet.



Joonis 13.Valemite muutmise vaade kasutajaliideses PG TaxRate välja näitel.



Joonis 14. Õpetus finantsandmete kaardistamise valemite koostamiseks.

4.4.3 Valemi muutmise testid

Töö autori jaoks on oluline mitte lasta ebakorrektseid valemeid salvestada ja anda administraatorile tagasiside valemi salvestamise kohta. Sellega saab muutja olla kindel, et kaardistamise valem on õigesti loodud.

Valemite kontrollimiseks kirjutas töö autor testid. Testid arvutavad kõik valemid (kaasaarvatud tagavara valemid) läbi test andmetega. Lisaks õigetele valemitele testitakse ka vigaseid valemeid. Kontrollitakse kas tagastab õiged vea teateid ja kas on võimalik

ebakorrektseid valemid muuta. Optimaalsuse mõttes ei arvutata valemeid läbi reaalsete andmetega, sest selleks tuleks kõik aastaaruanded alla laadida ja läbi arvutada.

Lisaks tuleb tähele panna, et ei valideerita välja nime korrektsust. Välja nime ei kontrollita jälle optimaalsuse seisukohast. Sellisel juhul oleks jälle vaja alla laadida kõik aastaaruanded ja vaadata kas ühes neist aasta aruandes on selline väli olemas – osades aasta aruannetes võib ühte kindlat välja sisaldada, aga järgmises mitte.

5 Lahenduse analüüs ja järeldused

Enne lahenduse tegemist soovis töö autor vaikimisi valemid üldisemalt defineerida. Näiteks, kui välja nimi sisaldab mingit sõna, siis on see otsitav finantsnäitaja. Paraku polnud nii võimalik lahendada, sest selliseid nimesid oli palju ja mõnikord oli see välja täpsustus ja teinekord oli otsitav väli teise nimega. Selline lahendus tähendab seda, et tulevikus tuleb aina rohkem ettevõtete valemite kaardistamist muuta.

Enne lahenduse tegemist oli töö autoril mitmeid häid ideid valemite muutmiseks kasutajaliideses. Näiteks oli mõttes, et koostada valemeid finantsandmeid lohistades. Seoses sellega, et andmete hulk oli nii suur ja valemi muutmise vaates polnud piisavalt ruumi, siis polnud see enam nii hea idee.

Selleks, et kasutajaliidesest tehtavad päringuid liiga aeglased poleks oli vaja info andmebaasi salvestada. Töö autori üllatuseks oli seda infot palju. Tihti tuli olukord, kus töö autor pidi jälle välja mõtlema, kuidas veel optimaalsemalt päringuid teha.

Töö autor on lõpplahendusega rahul järgmistel põhjustel:

- kasutajaliidese vaated sobivad kokku rakenduse üldise stiiliga;
- tagasüsteemi kood muutus loetavamaks;
- loodud lahendus vastas lõputöö alguses seatud nõuetele;
- töö autor tegi valemi muutmise võimalikult lihtsaks – valemi struktuuri ja õpetustega;
- peamised funktsionaalsused on testitud automaattestidega;
- lahendust ehitades polnud vaja uut tehnoloogiat kasutusele võtta ehk sai olemasolevat kasutatud.

Töö autor testis kasutajaliidese vaateid manuaalselt ja tagasüsteemi funktsionaalsust testiti automaatsete testidega. Mõned automaatsed testid olid juba varasemalt olemas, mis kinnitas ka, et varasemat loogikat ei muudetud. Uuele funktsionaalsusele lisas autor uued testid. Lisaks testis töö autor, et lahendus vastaks töö alguses seatud tingimustele.

Uued testid käivitatakse koos teiste testidega iga kord, kui keegi teeb koodi muudatuse. Sellega tagatakse see, et keegi töö autori loogikat kogemata ei muuda.

Valemite automatiseeritud kaardistamine oli rakenduse jaoks uus funktsionaalsus, mis lisab rakendusele lisandväärtust.

Edasised arendused sõltuvad administraatori tagasisidest loodud lahendusele ja juurdearenduse lisandväärtusest. Töö autori ideed edasiseks arenduseks on järgmised:

- valemis väljade valideerimine;
- vaikimisi valemite muutmine;
- luua vaade, mis näitab enne salvestamist, mis aastate finantsnäitajad muutuvad;
- kõikide väljade grupeerimine;
- proovida päringuid veel optimeerida.

Kindlasti eelnevalt nimetatud ideedele lisandub veel uusi ideid. Salvestatud andmeid saab hiljem analüüsida ja mille abil võibolla saab kaardistamist veel automatiseerida. Muidugi ideed sõltuvad prioriteetidest ja kui palju lisandväärtust rakenduse kasutajatele annab.

Töö autori loodud finantsandmete automatiseeritud kaardistamine on nüüd administraatorkasutajate jaoks kättesaadav ja valmis kasutamiseks.

6 Kokkuvõte

Lõputöö probleemiks oli, et andmeväljad on erineva nimega ja detailsusastmega ning selle muutmine koodis muutus tülikaks ja aeganõudvaks arendaja ning administraatori jaoks.

Lõputöö eesmärgiks oli automatiseerida finantsandmete kaardistamise. Töö autor täitis seatud eesmärgi, mille tulemusel saab administraator iseseisvalt ettevõtte finantsandmete kaardistust kontrollida ja muuta. Lisaks arendaja ei pea tegema koodi muudatusi, et kaardistus muuta.

Töö autor kasutas agiilset arendusmetoodikat finantsandmete automatiseeritud kaardistamise lahenduse ehitamisel. Ehk jagas lahenduse osadeks ja tegi uued funktsionaalsused osa kaupa kättesaadavaks.

Töö autor jagas finantsandmete automatiseeritud kaardistamise neljaks osaks:

- valemite kaardistamine;
- kõikide finantsandmete kuvamine;
- kasutatud valemite kuvamine;
- valemite muutmine.

Valemite kaardistamisel jagas valemid vaikimisi ja ettevõtte põhisteks valemiteks, lõi ühtse struktuuri valemite koostamiseks ja loogika, mis oskab valemite põhjal finantsnäitajaid arvutada.

Finantsandmete kontrollimiseks ja kaardistuse muutmiseks on vaja näha lisainformatsiooni. Selleks lõi töö autor vaated, et näha ettevõtte kõiki finantsandmeid ja kasutatud valemeid.

Finantsnäitajate kaardistamise valemite muutmise osas tegi töö autor kasutajaliidese vaate, kus on võimalik kohandada valemit oma soovi järgi. Samasse vaatesse lisas töö autor ka õpetuse valemite koostamise kohta. Vigaste valemite vältimiseks lisas töö autor valemi kontrolli, mis annab salvestamisel vastava teate.

Kõik eelnevalt nimetatud vaated kasutavad ühtset rakenduse stiili. Nii nad sarnanevad teiste vaadetega.

Lahenduse valideerimiseks kirjutas töö autor ühiktestid tagasüsteemi ja testis manuaalselt kasutajaliidese vaateid. Lisaks testis töö autor lõputöö alguses seatud testjuhtumeid, mille töö autori loodud lahendus ka edukalt läbis.

Seoses sellega, et andmeid oli palju pidi töö autor palju vaeva nägema koodi optimeerimisega. Esiteks pidi andmebaasi salvestada andmeid, et kasutajaliidese päringud oleksid kiiremad. Teiseks pidi lisama funktsionaalsuse, mis salvestab andmebaasi ühe korraga mitu dokumenti selleks, et sarnaste päringutega andmebaasi vähem koormata.

Loodud lahendus on kätte saadav administraatoritele ja valmis kasutamiseks.

Ideid edasiarenguks tekkis töö autoril palju, aga need ideed sõltuvad väga palju administraatori tagasisidest.

Kasutatud kirjandus

[1] XBRL International Inc., "The Standard for Reporting," [Online]. Available: https://www.xbrl.org/the-standard/what/the-standard-for-reporting/. [Accessed 22 04 2020].

[2] P. Hamack, J. Truzzolino and M. Savage, "How are U.S. governments spending YOUR tax dollars?," 20 04 2020. [Online]. Available: https://xbrl.us/xusnews/pov-2/. [Accessed 22 04 2020].

[3] Finantsinspektsioon, "Jõustus börsiemitentide finantsandmed paremini ligipääsetavaks muutev määrus," 18 06 2019. [Online]. Available: https://www.fi.ee/et/uudised/joustus-borsiemitentide-finantsandmed-paremini-ligipaasetavaks-muutev-maarus. [Accessed 22 04 2020].

[4] Vue.js, "Introduction," [Online]. Available: https://vuejs.org/v2/guide/. [Accessed 22 04 2020].

[5] Codersera, "Angular vs. React vs. Vue: A 2020 comparison," [Online]. Available: https://codersera.com/blog/angular-vs-react-vs-vue/. [Accessed 13 05 2020].

[6] https://golang.org/, "Frequently Asked Questions (FAQ)," [Online]. Available: https://golang.org/doc/faq. [Accessed 22 04 2020].

[7] aws.amazon.com, "Amazon DynamoDB," [Online]. Available: https://aws.amazon.com/dynamodb/. [Accessed 22 04 2020].

[8] GitHub Inc., "The State of the Octoverse," [Online]. Available: https://octoverse.github.com/. [Accessed 13 05 2020].

[9] B. Reinkensmeyer, "5 Best Free Stock Chart Websites for 2020," 05 04 2020. [Online]. Available: https://www.stocktrader.com/free-stock-charts/. [Accessed 22 04 2020].

[10] M. B. A. v. B. A. C. Kent Beck,"Agiilse tarkvaraarenduse manifesti põhimõtted," [Online]. Available: https://agilemanifesto.org/iso/et/principles.html.[Accessed 13 05 2020].

[11] A. Koutifaris, "Test Driven Development: what it is, and what it is not.," 2 07 2018. [Online]. Available: https://www.freecodecamp.org/news/test-driven-development-what-it-is-and-what-it-is-not-41fa6bca02a2/. [Accessed 13 05 2020].

[12] zdebeer99. [Online]. Available: https://github.com/zdebeer99/goexpression. [Accessed 22 04 2020].

[13] aws.amazon.com, "BatchWriteItem," [Online]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_BatchWriteItem.html. [Accessed 22 04 2020].