TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristjan Düüna 142653IAPB

PDF ARVETELT ANDMETE LUGEMISE AUTOMATISEERIMINE

bakalaureusetöö

Juhendaja: Tarvo Treier
MSc

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: Kristjan Düüna

25.05.2020

Annotatsioon

Lõputöö eesmärgiks on luua ettevõtte X näitel lahendus, mis aitaks arvete PDF-idest lugeda olulist informatsiooni ja seda konkreetse PDF-arvega seostada. Selle jaoks tutvuti esmalt olemasolevate protsessidega, kus on vaja PDF-arvelt informatsiooni lugeda. Seejärel analüüsiti juba olemasolevaid lahendusi ja lõpuks asuti lahendust arendama. Lahendus arendati valmis etappide kaupa, kus iga etapi lõpus anti ettevõttele kasutusse töötav lahendus. Iga etapp muutis mingil määral ettevõtte X protsesse lihtsamaks ja kiiremaks.

Lõpuks valmis eraldi mikroteenus, mis loeb PDF-arve pealt andmed ja tagastab need olemasolevasse CRM/ERP lahendusse. PDF-arve pealt andmete lugemiseks konverteeritakse kõigepealt PDF-arve tekstiks ja seejärel regulaaravaldiste abil eraldatakse sealt oluline informatsioon.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 6 peatükki, 23 joonist, 0 tabelit.

Abstract

Automation of Data Extraction from PDF Invoices

The purpose of this thesis is to implement a solution to extract data from PDF invoices and to match it with the existing documents. The provided solution is created for a specific company and is currently in active use.

First part of thesis focuses on inspecting the existing solution to learn about the business processes currently in use. Second part analyzes the preexisting solutions found on the internet. Third part describes the development of the resulting service. Last part analyzes the created software.

Solution was done using agile development and MVP methods. This way client could get an initial usable product very quickly and later get updates for it. Microservices architecture allowed to easily use technologies that were not compatible with the preexisting codebase. Project also used CI/CD approach that automatically tests and builds the code after push to the git repository. Final service is in a Docker container to make it easy and fast to deploy.

The result of this thesis was a separate microservice that can extract information from PDF invoices and forwards the result to the existing CRM/ERP solution. To extract information from PDF, it is firstly converted to text using Apache PDFBox library. Then the important data is extracted using written rules and regular expression matching. Finally, the extracted information is inserted into CRM/ERP solution and linked with the PDF invoice.

The thesis is in Estonian and contains 40 pages of text, 6 chapters, 23 figures, 0 tables.

Lühendite ja mõistete sõnastik

AS-IS	Hetkeolukord
API	*Application programming interface*, rakendusliides
BDOC	ZIP-konteiner, mis sisaldab allkirjastatud faile, allkirju ja juhtinfot
CI/CD	*Continuous integration and continuous delivery*, pidevintegratsioon ja pidev edastamine
CRM	*Customer relationship management*, kliendisuhete juhtimine
CSV	*Comma-separated values*, tekstifail, kus erinevad väärtused on omavahel eraldatud komaga
Digiteerimine	Füüsilise objekti või analoogmaterjali digitaalsele kujule viimine
DLL	*Dynamic-link library*, dünaamiline teegifail
Docker	Arvutiprogramm, mis teostab operatsioonisüsteemi tasemel virtualiseerimist ehk konteineriseerimist
ERP	*Enterprise resource planning*, ettevõtte ressursside planeerimine
FTP	*File Transfer Protocol*, failiedastusprotokoll
Hardcoded	Andmed programmis, mille muutmiseks on vaja muuta programmikoodi
Interop	Interoperability, koostalitlusvõime
Kliendi arve	Arved toodete ja teenuste eest, mida klient tarbib teiste teenusepakkujate juures, kellega ettevõttel X puudub otseleping
MVP	*Minimum viable product*, minimaalne töötav toode
OCR	*Optical character recognition*, optiline märgituvastus
Partneri arve	Partnerite arved toodete ja teenuste eest, mida ettevõte X vahendab enda klientidele
PDF	*Portable Document Format*, PostScript-il põhinev arvuti riist- ja tarkvaraplatvormist sõltumatu elektrooniliste dokumentide vorming
Sõsarettevõte	Lähedane või sama tüüpi ettevõte, kellel on ühine vahetu või kaudne otseinvesteerija
Taustatöö	Perioodiliselt käivitatav programmi osa
ZIP	Arhiveerimise failiformaat
TO-BE	Tulevikuolukord

Sisukord

Jooniste loetelu

1 Sissejuhatus

Eestis on viimasel ajal olnud palju juttu e-arvetest ja e-kviitungitest. Turul on äppe, millega saab pildistada tšekki, tuvastades sellel olevad andmed automaatselt ning saates edasi need otse raamatupidamisse. Hoolimata suurtest arengutest muuta arveldus paberivabaks, leiab veel paljudes ettevõtetes virnade viisi pabereid või PDF-ide ja skaneeritud dokumentide katalooge, millel olevaid andmeid sisestatakse enda süsteemidesse käsitsi. Käsitsi sisestamise vajaduse põhjus pole alati see, et ettevõtted ei tahaks e-arveid ja e-kviitungeid kasutusele võtta. Näiteks võib hetkel käsitsi töötlemise vajaduse tingida olukord, kus arve väljastajaks on välisriigi ettevõte, arve saaja ja maksja on erinevad, arved on vaja edastada partnerile tema soovitud viisil.

Käesolevas töös näitena toodud ettevõte X tegeleb kliendi arvete faktoorimisega ja kliendi eest maksuametile käibemaksu tagastustaotluste esitamisega teises Euroopa Liidu riigis tehtud ostudelt, mis on seotud kliendi äritegevusega. Mõlema teenuse sisendiks on arved. Arvete faktoorimisega soovib klient saada raha kätte enne arvel toodud maksetähtaega ja selleks müüb ta arvega kaasneva nõude ettevõttele X. Ettevõte X peab vormistama vastava lepingu ja seejärel jääb ise arve eest laekumist ootama sellelt ettevõttelt, kellele arve väljastati. Käibemaksu tagastustaotluste korral on arveteks tavaliselt klientidele esitatud arved välisriigis tarbitud kütuse ja teemaksu eest. Kui kõik nende arved välja trükkida, millega ettevõte X tegelema peab ühes kuus, siis võib lehekülgede arv minna kümnetesse tuhandetesse.

Antud bakalaureusetöö autor puutus ettevõttega X kokku nende CRM (*customer relationship management*) ja ERP (*enterprise resource planning*) lahenduse arendajana. Esimesel kokkupuutel oli ettevõttel erinevatel põhjustel kujunenud aastate jooksul välja protsessid, mille käigus trükiti paljud nendest kümnetest tuhandetest lehekülgedest iga kuu välja. Osad arved saadeti ettevõttesse X paberkujul.

Lõputöö käigus loodava lahenduse puhul on autor olnud arhitekti ja arendaja rollis alates 2018. aasta juunikuust.

1.1 Eesmärk

Käesoleva töö eesmärgiks on luua ettevõtte X näitel lahendus, mis aitaks arvete PDF-idest lugeda olulist informatsiooni ja seda konkreetse PDF-iga seostada.

Antud eesmärgi täitmiseks tuleb lahendada järgmised ülesanded:

- Tutvuda faktooringu ja käibemaksu olemasoleva protsessiga.
- Analüüsida erinevaid PDF-e ja nende lugemise võimalusi.
- Arendada meetod PDF-i parsimiseks ja sealt oluliste andmete välja küsimiseks.
- Kasutajaliides disainida ja arendada.
- Testimine, pidev integratsioon (*continuous integration*).

1.2 Ülesehitus

Bakalaureusetöö teises peatükis antakse ülevaade taustast, lühitutvustus ettevõttest X ja ülevaate arvete faktoorimise ja käibemaksu tagastamise protsessidest.

Kolmandas peatükis analüüsitakse, kuidas need protsessid toimuvad ettevõttes X. Peale seda uuritakse ja analüüsitakse juba turul olevaid lahendusi. Kõike seda põhjusel, et oleks ülevaade sellest, mida hakatakse asendama ja leida ideid olemasolevatest lahendustest.

Neljandas peatükis käsitletakse lahenduse realiseerimist. Arendus toimus etappide kaupa ja iga etapi lõpus antakse ülevaade, kuidas on need olemasolevat protsessi mõjutanud.

Viiendas peatükis vaadatakse loodud lahendusele tagasi. Analüüstakse ja hinnatakse loodud lahendust.

Viimases peatükis antakse ülevaade tehtud tööst.

2 Taust

Antud peatükis tutvustatakse ettevõtet X. Seejärel kirjeldatakse arvete faktoorimise ja piiriülese käibemaksu tagastamise protsessi. Eesmärgiks on anda ülevaade taustast, et töö sisu oleks selgemini mõistetav.

2.1 Ettevõttest

Ettevõtte X puhul on tegemist nelja erineva sõsarettevõttega, kes tegelevad kolmes erinevas Euroopa riigis. Antud lõputöö raames vaadatakse neid kui ühte ettevõtet X.

Ettevõte X tegeleb kliendi arvete faktoorimisega ja kliendi eest maksuametile käibemaksu tagastustaotluste esitamisega teises EL riigis tehtud ostudelt, mis on seotud kliendi äritegevusega. Mõlema teenuse sisendiks on arved.

2.2 Protsessist

Järgnevalt antakse ülevaade arvete faktoorimise ja käibemaksu tagastamise protsessidest.

2.2.1 Arvete faktoorimine

Arvete faktooring on finantsteenus, mille puhul sisuliselt müüakse arve faktooringupakkujale. Arve müümisel faktooringupakkujale saadakse teenustasu eest kohe kätte arve summa. Seejärel maksab ostja ise arve summa hoopis faktooringupakkujale. Arvete faktooringu kasutamise tingib asjaolu, et ettevõtetes on tavaliselt arvete maksetähtajad pool kuud kuni mitu kuud. Selleks, et arve eest raha kohe kätte saada, kasutatakse arvete faktoorimise teenust. See võimaldab vabanenud käibevahendeid investeerida äritegevusse kiiremini [1].

Faktooringuprotsessis on kolm osapoolt: müüja, ostja ja faktooringupakkuja. Esmalt sõlmib müüja faktooringupakkujaga lepingu selleks, et hakata kasutama arvete faktoorimise teenust. Edaspidi lisab müüja enda väljastatavatele arvetele märke, et

tegemist on faktooritud arvega ja arve peale pannakse faktooringupakkuja pangakonto andmed. Seejärel edastab müüja need arved ka faktooringupakkujale. Faktooringupakkuja veendub, et arve alusel on reaalselt teenust või kaupa müüdud ja ostja on võimeline selle arve tasuma. Peale seda makstakse kliendile välja osa arvest, enamasti jääb see summa 70–90% vahemikku. Edasi jääb faktooringupakkuja ootama arve maksmist ostja poolt. Kui arve on makstud, makstakse ka ülejäänud summa müüjale välja. Kõigest sellest arvestatakse maha teenustasud, käitlustasud ja muu, mis lepingus kokku lepiti [1]. Kogu see protsess on lihtsustatud kujul ära toodud ka joonisel 1.

Joonis 1. Faktooringuprotsess.

2.2.2 Piiriülene käibemaksu tagastus

Käibemaksu tagastamise protsessi kirjeldamiseks on kasutatud Maksu- ja Tolliameti poolt koostatud dokumenti „Piiriülene käibemaksu tagastamise süsteem“ [2]. Piiriülene käibemaksu tagastamise protsess Euroopa Liidus on paika pandud Euroopa Liidu poolt ja on seetõttu sarnane igas Euroopa Liidu liikmesriigis.

Käibemaksu saab tagasi taotleda vaid käibemaksukohuslane ettevõte. Enne, kui käibemaksu tagasi taotlema saab hakata, peab olema soetatud kaupade ja teenuste kohta arveid. Arvete alusel tuleb esitada taotlus ettevõtte asukohariigi maksuametile. Taotluse tegemisel tuleb esmalt otsustada, mis riigi ja mis perioodi kohta taotlust soovitakse teha. Minimaalne taotlusperiood on kolm kuud ja maksimaalne üks aasta. Selleks tuleb kokku koguda kõik selles riigis tehtud tehingute arved antud perioodi kohta. Need tuleb esitada kõik koos ühe taotlusena.

Taotluse tegemisel tuleb edastada taotluse esitaja andmed, taotluse üldandmed ning pangakonto andmed, kuhu tagastatav raha kanda. Lisaks tuleb iga dokumendi kohta veel esitada täpsemad andmed. Oluline on lisaks dokumendi üldandmetele veel käibemaksu summa ja arve alusel soetatud kaupade ning saadud teenuste laad. Nende kirjeldamiseks on määratud konkreetsed koodid, mis on paika pandud Euroopa Liidu Nõukogu direktiiviga 2008/9/EÜ artiklis 9 [3, lk 6]. Taotluse juurde tuleb lisada ka kõikide arvete dokumendid. Need võivad olla PDF-faili kujul, pildina või skaneeritult. Ühe taotluse kohta võib edastada ainult ühe faili, seega tuleb need koondada kas ZIP-failiks või kokku üheks PDF-failiks.

Pärast taotluse esitamist asukohariigi maksuametile kontrollitakse see nende poolt üle ja saadetakse edasi tagastamisriigi maksuametile. Nemad menetlevad taotlust vastavalt kohaliku riigi seadusandlusele ja tagastavad ettenähtud summa.

3 Analüüs

Käesolevas peatükis uuritakse, kuidas toimib hetkel arvete faktoorimine ja käibemaksu tagastamine ettevõttes X. Eesmärgiks on viia ennast kurssi hetkel kasutusel oleva protsessiga, et teaks mida asendama hakatakse. Oluline on üles leida nõuded, mis peaksid kindlasti paika jääma. Samuti tuleks leida ideid olemasolevast protsessist, mida võtta kasutusele loodavas lahenduses.

Peale esialgse analüüsi tegemist uuritakse juba turul olevaid lahendusi. Eesmärgiks on leida, kas mõni juba sobiks olemasoleval kujul. Kui mitte, leitakse nendest ideid, mida tahetakse tehtavas lahenduses ära kasutada. Seejärel analüüstakse erinevaid PDF-arveid, mida kliendid saadavad ettevõttesse X.

Lõpetuseks pannakse analüüsi põhjal paika nõuded, millele loodav lahendus peab vastama.

3.1 Olemasolevad protsessid ettevõttes (AS-IS)

Aastate jooksul on välja kujunenud kindlad protsessid, kuidas toimub ettevõttes X arvete faktoorimine ja käibemaksu tagastuse taotluste tegemine. Esmalt analüüsitakse mõlemat protsessi eraldiseisvana. Seejärel leitakse üldistusi impordi osas ning kõige lõpuks pannakse paika esialgsed analüüsitulemused. Eesmärgiks on leida ideid, et muuta olemasolevaid protsesse paremaks.

3.1.1 Arvete faktoorimine

Arvete faktooringu puhul tehakse kõigepealt kliendiga baasleping, kelle arveid faktoorima hakatakse. Peale lepingu sõlmimist hakkab klient saatma faktoorimist vajavaid arveid ettevõttesse X. Kliendilt saadud arvetelt vaadatakse välja olulised andmed ja sisestatakse need olemasolevasse CRM/ERP lahendusse.

Kui arved on CRM/ERP lahendusse sisestatud, tehakse kliendiga konkreetne leping just nende arvete faktoorimiseks. Selles lepingus pannakse paika, mis arveid faktoorima hakatakse, samuti määratakse ära lepingutasud. Peale lepingu kahepoolset

allkirjastamist makstakse kliendile vastavalt lepingus määratud summa ning jäädakse ootama nende arvete eest raha laekumist.

3.1.2 Käibemaksu tagastamine

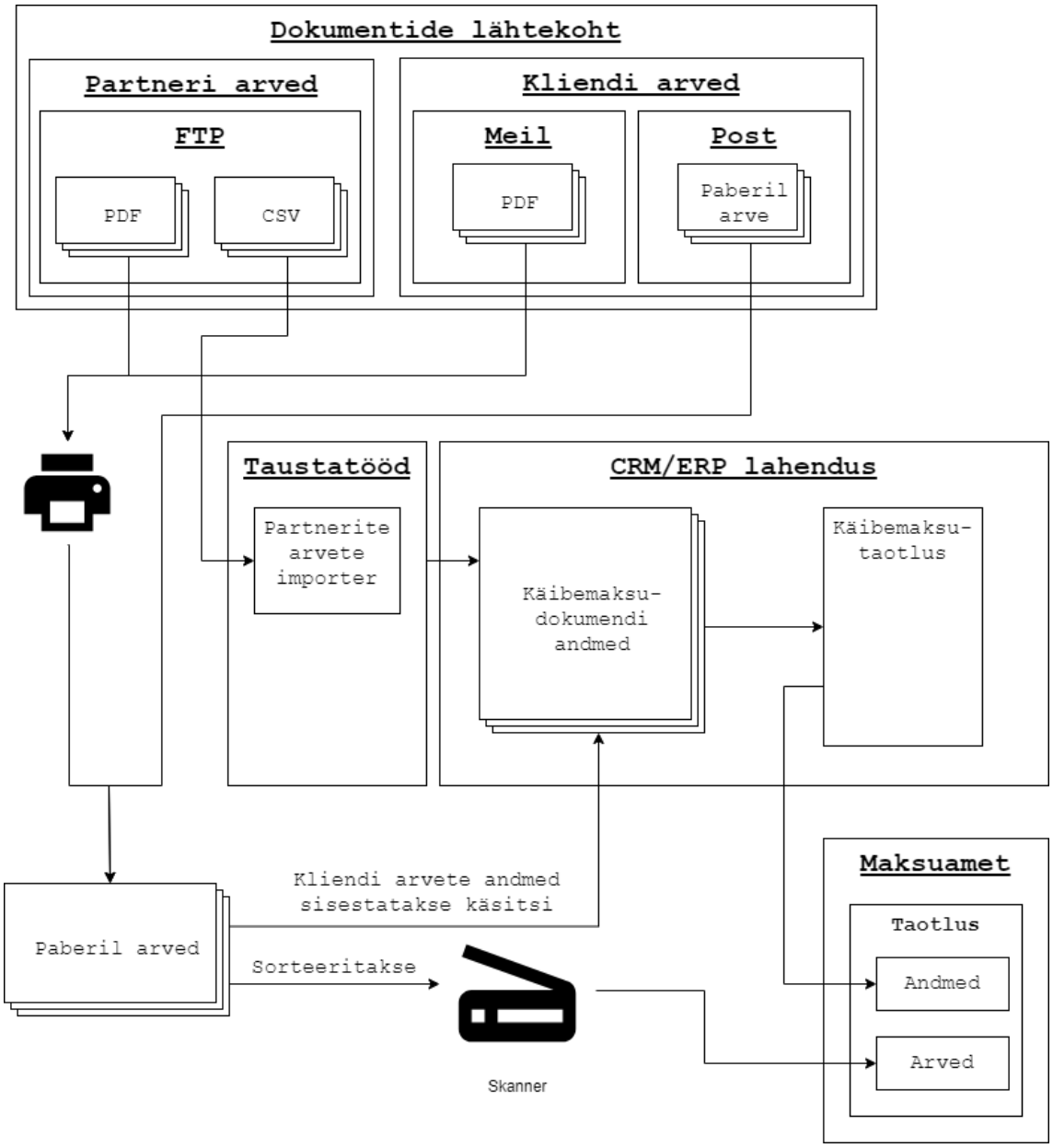
Visuaalne ülevaade käibemaksu taotlemise protsessist on ära toodud joonisel 2. Üldiselt on olemas kahte tüüpi arveid: ühed on partnerite arved ja teised kliendi arved. Partnerite arvete puhul on tegemist arvetega, mis saadakse otse partnerilt. Kliendil on leping ettevõttega X, kes vahendab partneri teenuseid. Partner esitab arve ettevõttele X, kes omakorda esitab selle alusel kliendile arve. Partner laeb enda arved FTP serverisse, kust saab need kätte. Iga arve kohta on olemas PDF-fail ja CSV-fail. PDF sisaldab arvet ennast ja CSV-failis on vastava arve andmed masinloetaval kujul. CSV-failide põhjal on loodud taustatöö, mis võtab nendest failidest andmed ja loob nende põhjal CRM/ERP lahendusse käibemaksudokumendid.

Kliendi arvete puhul on tegemist arvetega, mis tulevad ettevõttesse X enda klientide poolt. Nende puhul on tegemist arvetega teenuste eest, mida klient tarbib teiste teenusepakkujate juures, kellega ettevõttel X ise otselepingut pole. Klientide poolt tuleb ettevõttesse X lisaks arvetele ka tšekke, enamasti on nendel kas parkimise või teemaksu tasud.

Digitaalsel kujul ettevõttesse X saabunud arved prinditakse kõik välja. Seda põhjusel, et partnerite poolt tulevad koondarved, kus on ühe kliendi kõikide riikide tarbitud tooted ja teenused peal. Need on küll eraldi riikide kaupa, kuid koos ühes PDF-failis. Nende lihtsamaks tükeldamiseks ja sorteerimiseks prinditakse need välja. Lõpuks tekib iga kliendi arvete jaoks eraldi hunnik, kus lisaks on need arved ka riikide kaupa eraldatud ning sorteeritud vastavalt maksuameti poolt nõutud järjekorrale. Välja prinditud klientide arvete põhjal sisestatakse CRM/ERP lahendusse arvete andmed käsitsi.

Kui jõuab kätte aeg teha kliendile käibemaksu tagastuse taotlus, korjab CRM/ERP lahendus vajalikud käibemaksudokumendi andmed välja. Seda tehakse vastavalt kliendile, riigile ja taotlemise perioodile. Seejärel võetakse ette vastava kliendi arved paberi peal ja kontrollitakse üle, kas kõik taotluses olevad arved on ikka paberi peal olemas ja vastupidi. Kui kõik on korras, skaneeritakse need arvutisse üheks PDF-failiks. See PDF-fail pannakse maksuametisse esitatavale taotlusele kaasa. Põhiandmed taotluse jaoks leiab CRM/ERP lahendus ise kokku taotluses olevates käibemaksudokumentidest.

Edasi jäädakse ootama maksuameti otsust ja raha laekumist. Kui maksuametilt on raha laekunud, siis makstakse see kliendile välja.



Joonis 2. Käibemaksu tagastamise protsess ettevõttes X.

3.1.3 Impordi osa sarnasused/erinevused

Antud lõputöös vaadeldakse eelkõige PDF-arvetelt andmete automaatset lugemist. Seetõttu on protsessis oluline just PDF-arvete ülesehitus. Järgnevalt tuuakse välja arvete impordi osa sarnasused.

Arvete faktoorimise ja käibemaksu tagastamise protsessi puhul on sisendiks arved, millelt tuleb lugeda välja vajalik informatsioon ja sisestada koos andmetega olemasolevasse CRM/ERP lahendusse. Arvetelt loetava informatsiooni võib jagada kaheks. Esiteks on arve üldandmed, need on olemas mõlemal arvel. Nendeks on arve number, arve kuupäev, arve kogusummad, kellele arve väljastati, kes arve väljastas. Nende lugemine toimub mõlema arve puhul sama moodi.

Teist tüüpi andmed on toodete ja teenuste andmed arve peal, mida on vastavalt erinevate toodete ja teenuste tarbimise kogusele. Nende andmete lugemine on oluline ainult käibemaksu tagastamise protsessi juures, kuna käibemaksu tagastamise korral tuleb määrata taotluse juures ära, mis tüüpi tooted ja teenused on taotletava arve peal. Samuti on tooteid ja teenuseid, mille pealt käibemaksu tagasi ei saa. Sellisel juhul tuleb need taotluse peal olevast taotletavast summast välja jätta. Faktooritavate arvete puhul ei ole see informatsioon oluline.

Sellest selgub, et impordi poole pealt on faktooringuarved käibemaksuarvete alamosa. Käibemaksuarvetelt on vaja rohkem andmeid kätte saada ja sellega kaetakse ära kõik need andmed, mis on vajalikud faktooringuarvete pealt kätte saada. Seetõttu lähtutakse lahenduse käsitlemisel põhiliselt käibemaksu tagasi taotlemise protsessist.

3.1.4 Soovitud muudatused protsessis (TO-BE)

Esmane ja võimalik, et ka kõige olulisem nõue on, et ei peaks enam ühtegi arvet välja printima. Kogu arvete tükeldamine, sorteerimine ja kokkupanemine peab olema süsteemi poolt tehtav automaatselt või poolautomaatselt.

Klientide poolt peaksid tulema kõik arved digitaalselt meili peale. Analüüsis nägime, et klientide arved on pärit teenusepakkujatelt, kes ei saada neid kindlasti postiga kliendile kontorisse. Pigem võtab klient need iseteenindusest või meili pealt ja prindib need ise välja. Seetõttu oleks oluline kliendile teha teavitustööd, et ta saadaks kõik arved jooksvalt ettevõttesse X digitaalsel kujul meili peale.

Arveid peaks saama kogu aeg üle vaadata, nii partnerite kui ka kliendi poolt saadetud arveid. Tšekid ei ole antud olukorras olulised, kuna nende hulk on võrreldes kõige muuga väga väike. Käibemaksudokumendi juurest peaks saama vastava PDF-dokumendi otse avada.

3.2 Olemasolevad lahendused

Selles peatükis uuritakse, kas juba leidub lahendusi, mida saaks ettevõttes X kasutusele võtta. Samuti analüüsitakse olemasolevaid lahendusi ja proovitakse leida ideid, mida loodavas lahenduses ära kasutada. Seda juhul, kui ükski neist olemasolevatest lahendustest ei sobi kasutamiseks ettevõttes X. Järgnevalt vaadeldakse kolme leitud lahendust, mis tundusid täitvat vajaminevat eesmärki

3.2.1 CostPocket

CostPocket-i puhul on tegemist lahendusega, mis aitab ettevõttel kokku koguda ning digiteerida kõik kviitungid ja arved ning edastada need ettevõtte raamatupidamis-tarkvarasse [4]. Tegemist on Eesti ettevõttega, mis asustati märtsis 2016 [5]. Tootega tuldi turule 2017. aasta alguses [5].

Antud lahenduse tööpõhimõtteks on see, et peale kviitungi saamist tehakse sellest nutitelefoniga pilt. Toetatud on Android ja iOS operatsioonisüsteemid. Peale pildistamist tuvastatakse kviitungilt OCR-i (*optical character recognition*) abil olulised andmed ja muudetakse need masinloetavale kujule. Kui midagi ei suudeta automaatselt leida või loeti valesti välja, on võimalus kasutajal andmeid täiendada ja muuta. Kui andmete õigsuses ollakse kindlad, kinnitatakse kviitung ära ja laaditakse see süsteemi üles. Võimalik on lisada liidestus ka raamatupidamistarkvaraga. Sellisel juhul jõuab sisestatud kviitung koheselt raamatupidamistarkvarasse. Lahenduse peamiseks eesmärgiks on muuta ettevõtte kuludokumentide haldamine lihtsamaks ja kiiremaks [4].

Antud lahenduse on mõeldud eelkõige firma enda kuludokumentide haldamiseks. Ettevõttes X on oluline töödelda teiste firmade arveid. Kuigi mingi osa on kviitungeid, siis põhiline osa on ikkagi PDF-arved. Selliste arvete jaoks antud süsteem ei sobi. Samuti kviitungite eraldi pildistamine oleks palju ajamahukam, kui on praegune lahendus. Hetkel pannakse kõik ühe kliendi ja samas riigis tarbitud teenuste kviitungid korraga skanneri vahele ja skaneeritakse korraga sisse. Samuti ei oska CostPocket kviitungite pealt lugeda välja seal peal olevaid tooteid, kuid need on käibemaksu tagastamise puhul olulised. Sellisel juhul peaks ikkagi kasutaja kõik üle vaatama. Antud lahendusel on olemas konkreetne sihtrühm, kuid ettevõttes X kasutamiseks see lahendus ei sobi.

Kuna käibemaksu tagastamise jaoks tuuakse ettevõttesse X ka vähesel määral kviitungeid, siis antud lahendusest saaks kaasa võtta OCR-i kasutamise andmete lugemiseks, kuid loodavas lahenduses peab olema võimalus kviitungid korraga sisse skaneerida. Seejärel peaks tarkvara kviitungid eraldi tükeldama ja lugema välja olulised andmed nende pealt ja sisestama CRM/ERP lahendusse.

3.2.2 Docsumo

Docsumo on andmete sisestamise ja dokumente sisaldava tööprotsessi automatiseerimise lahendus. Lahenduse eesmärgiks on igat tüüpi arvetelt automaatselt oluliste andmete kättesaamine, olenemata dokumendi keelest ja struktuurist. Antud lahenduse juures ei ole vaja eraldi malli seadistada ja andmete lugemistäpsus peaks olema võrreldav sellega, kui inimene sisestab dokumendi põhjal andmeid käsitsi süsteemi [6].

Dokumendist teksti kättesaamiseks kasutatakse intelligentset OCR-i ja sügavaid närvivõrke (*deep neural network*). Kui andmed on välja loetud, tuleb need manuaalselt üle vaadata ja ära kinnitada. Docsumo kuvab ülevaatamise faasis kasutajale välja kõik andmed võtme-väärtuse paaridena. Kui kuskil on vigu, siis saab need kohe ära parandada. Kui midagi jäi lugemata, saab kohe dokumendist valida õige väärtuse. Peale andmete kinnitamist on dokumentide informatsioon kättesaadav CSV-failina või kui lisada liidestus olemasoleva süsteemiga API vahendusel, edastatakse need andmed sinna [6], [7].

Antud süsteemi probleemideks on see, et iga dokument tuleb käsitsi üle vaadata ja ära kinnitada. Alles peale seda liiguksid dokumendi andmed edasi ettevõtte X süsteemi. Ettevõttes X on hiljem vaja veel külge panna toodete põhjal teenused ja ka need peaksid olema kasutaja poolt ülevaadatavad. Kasutaja peaks veenduma, et külge minevad teenused on korrektsed. Seepärast tekitaks antud lahendus topelt töö, kuna tuleks kahes erinevas kohas kinnitada PDF-arvelt saadud andmete õigsust.

3.2.3 Docparser

Docparser-i puhul on tegemist lahendusega, mille eesmärgiks on tööprotsesside automatiseerimine just andmete sisestamise poole pealt. Antud lahendusega on võimalik erinevatelt dokumentidelt kätte saada andmeid. Olgu selleks PDF-arve, skaneeritud arve, leping, töökäsk või panga väljavõte [8].

Dokumendist andmete välja lugemiseks kasutatakse malli põhist lähenemist. Igale dokumenditüübile luuakse eraldi kataloog, kus määratakse ära, mis malli kasutatakse sinna kataloogi pandud failidelt andmete lugemiseks. Malli põhjal tehakse kindlaks, mis andmed on dokumendi peal olulised ja mille alusel neid otsima peaks ning mis kujul need on. Erinevate andmete üles leidmiseks kasutatakse mustrite ja märksõnade põhist tuvastamist. Esmalt leitakse üles dokumendist kõik väljad, mis näevad välja nagu leidmist vajav andmeväli. Näiteks arve kuupäeva leidmiseks leitakse dokumendist üles kõik kuupäevad. Kui kandidaatandmed on üles leitud, siis vaadatakse iga välja juures olevaid märksõnu ja selle põhjal leitakse tõenäosus, kui hästi mingi tulemus võiks olla otsitav väärtus. Valitakse see väärtus, mis sobib kõige suurema tõenäosusega. Kui vaikimisi otsitakse kogu dokumendi pealt, siis vajadusel saab ka määrata konkreetse piirkonna, kus otsitav väärtus peaks paiknema [8].

Antud lahenduse puuduseks on see, et ei ole automaatset dokumenditüübi tuvastamist. Dokument tuleb üles laadida konkreetse malli alla ja edasi kasutatakse seda konkreetset malli andmete sõelumiseks. Ettevõttesse X saadab klient enda kõik arved ühe meiliga, kus on koos erinevate teenusepakkujate arved. Antud lahendust kasutades tuleks ikkagi kõik need arved esmalt ära sorteerida teenusepakkujate kaupa.

3.3 Erinevate kliendi PDF-arvete analüüs

Lisades 1-2 on toodud näiteid kliendi PDF-arvetest. Konfidentsiaalsusest lähtuvalt on arvete andmeid muudetud ja firma nimena on kasutatud Näidis firma OÜ. Arve väljastaja andmed on jäänud muutmata, kuna tegemist on tuntud firmadega ja need andmed on vabalt kätte saadavad.

Arveid saab jagada laias laastus kolme kategooriasse:

1. Väljastatakse iga tehingu kohta.
2. Väljastatakse kindla perioodi kohta ühes riigis.
3. Väljastatakse kindla perioodi kohta, kuid kõikides erinevates riikides tarbitud tooted ja teenused on ühe arve peal.

Esimese ja teise kategooria puhul vastab üks PDF-arve ühele käibemaksudokumendile. Esimesse kategooriasse klassifitseeruv arve on näiteks Neste arve, näide toodud lisas 1.

Kolmanda kategooria puhul tuleb sarnaselt partnerite arvetele need dokumendid ära tükeldada. Nende ülesehitus on sarnane partnerite arvetega, kuid nende pealt tuleb lisaks tükeldamisele lugeda ka andmed välja. Selline on näiteks Circle K arve, näide toodud lisas 2. Esimesel lehel on sarnaselt partneri arvetele toodud erinevates riikides tarbitud toodete ja teenuste koondsummad. Järgmistel lehekülgedel on riikide kaupa neis riikides tarbitud teenused.

3.4 Analüüsi tulemus

Lõpetuseks kokkuvõte nõuetest, millele loodud lahendus vastama peab:

- Peab saama ette anda mitu faili korraga.
- Peab toetama kokkupakitud failiformaate ZIP ja BDOC.
- Lahenduse peab tuvastama arve tüübi.
- Enne süsteemi salvestamist on olemas eelvaade ja võimalus parandada andmeid.
- Lahendus peab töötama nii käibemaksudokumentide kui ka faktooritavate arvete puhul.

4 Realisatsioon

Käesolev peatükk käsitleb loodud lahenduse järk–järgulist realiseerimist. Kuna tegemist on päris projektiga, päriselt inimesed töötavad, siis on oluline jõuda töötava lahenduseni võimalikult kiiresti. Seetõttu sai aluseks võetud MVP (*minimum viable product*) lähenemine. Vastavalt MVP lähenemisele on mõistlik teha valmis minimaalselt töötav lahendus ja anda see töötajate kasutusse [9]. Kõik, mis suudetakse kohe ja kiiresti ära teha, on otsene võit. Arendamise järjekord on valitud vastavalt sellele, mis tekitaks ettevõttele X kõige kiiremini kõige suurema kasu. Samuti on suur roll tagasisidel, mida saab palju kiiremini, kui anda kätte mingi osa kogulahendusest võimalikult kiiresti. Seejärel saab seda tagasisidet juba edasises arenduses arvesse võtta. Samuti on vähem, mida peab muutma hakkama, võrreldes sellega, kui anda kätte täielikult valmis arendatud tükk.

Esimese etapina arendati valmis partnerite arvete automaatne sidumine juba sisestatud käibemaksudokumentidega. Seda põhjusel, et partnerite arved on peamiseks põhjuseks, miks kõik arved välja prinditakse. Samuti on partnerite arveid kogumahust kõige suurem hulk. Esimese iteratsiooni lõpuks valmib lahendus, kus enam ei pea ühtegi arvet välja printima. Kliendi toodud arvete sisestamiseks luuakse eraldi võimalus, kuidas PDF-arveid käsitsi sisestatud käibemaksudokumendiga siduda.

Teise etapina luuakse lahendus, et saaks ka kliendi saadetud PDF-arveid automaatselt CRM/ERP lahendusse sisse. Seal tuleb andmed PDF-arve pealt kätte saada ja sisestada CMR/ERP lahendusse. Siin on oluline luua ka kasutajaliides, kust kohast saaks neid sisestama hakata ja kasutaja saaks ka enne lõplikku sisestamist veenduda andmete õigsuses.

Viimase etapina viiakse loodud lahendus põhiprogrammist eraldi mikroteenuse peale. Kasutusele võetakse CI/CD, Docker ja automaattestid.

4.1 Ettevalmistused paberarvetest loobumiseks

Esimeseks arenduseks valiti partnerite arvete sidumine juba taustatöö poolt CRM/ERP lahendusse sisestatud käibemaksudokumentidega. Seda põhjusel, et neid on lehekülgede arvu poolest kõige rohkem, kokku umbes 30 000 lehekülge, ja need kõik prinditakse välja. Esimeseks sooviks on loobuda PDF-arvete väljaprintimisest.

Partnerite arvete puhul on tegemist PDF-arvetega, mis tulevad ettevõttesse X otse partnerite käest. Klientidel on leping ettevõttega X, kes vahendab partnerite teenuseid.

4.1.1 Partnerite PDF-arvete sidumine käibemaksu dokumendiga

Partnerite arvete puhul on vaja PDF-dokument ära tükeldada riikide kaupa ja siduda olemasoleva käibemaksudokumendiga, mis on juba sisestatud taustatöö poolt automaatselt CRM/ERP lahendusse.

Olemasolevate partnerite arved on kõik samasuguse ülesehitusega. Esimesel lehel on koondülevaade, kus on iga riigi kohta kirjas, mis summade eest selles riigis teenuseid tarbiti. Järgmistel lehekülgedel on riikide kaupa konkreetselt selle riigi arve selles riigis tarbitud teenuste kohta. Iga riigi kohta on vastavalt vajadusele lehekülgi. Ka need on samasuguse ülesehitusega. Iga leheküljel on päis, kus on kirjas üldinfo ja edasi tuleb toodete ja teenuste informatsioon.

Partnerite arvete tükeldamiseks on oluline leheküljel sisalduv arve number. Selleks loetakse sama arve numbriga järjest leheküljed ja liidetakse need kokku üheks PDF-iks. Saadud PDF pannakse olemasolevale käibemaksudokumendile külge. Selleks loodi andmebaasi lisaveerg, kuhu läheb kirja dokumendi asukoht andmekandjal. Lihtsustamaks failide haldamist, liigutatakse enne PDF ettenähtud kausta võrgukettal.

4.1.2 Käsitsi PDF-arve sidumine käibemaksudokumendiga

Kuna nüüd on osad PDF-id seostatud käibemaksudokumentidega, tuleks ka teistele PDF-arvetele tekitada võimalus nende sidumiseks. Vastasel juhul tuleks ikkagi käsitsi paika panna arvete järjekord ja need kuidagi kokku liita üheks suureks koond-PDF-iks. Samas saadetakse klientide poolt paljude erinevate teenusepakkujate arveid. Et kõiki neid saaks kohe alguses automaatselt sisestada, oleks arendamisele kuluv aeg väga pikk. Mõistlikum on suuremate teenusepakkujate arveid hakata alguses automaatselt sisestama ja hiljem vastavalt vajadusele tegeleda väiksematega. Umbes 10 kõige

suuremat teenusepakkujat katavad ära 80% arvetest. Mõne teenusepakkuja arveid tuleb kuus alla kümne. Neid oleks mõistlikum esialgu käsitsi sisestada, kuni mahud kasvama hakkavad. Käsitsi PDF-i lisamiseks sai loodud olemasolevasse kasutajaliidesesse eraldi võimalus. Valida tuleb sobiv PDF, mida tahetakse käibemaksudokumendile külge panna. Valitud dokumendist tehakse koopia õigesse kausta ja lisatakse viide andmebaasi, käibemaksudokumendi külge.

4.1.3 PDF-ide mahu vähendamine

Käibemaksutaotluse esitamisel ei tohi esitatava faili maht ületada 5 MB [2, lk 5]. Kui varasemalt skaneeriti kõik ühe käibemaksutaotluse arved korraga sisse, üldjuhul must-valgena, vajadusel sai skaneerimise kvaliteeti vähendada, kui arveid oli rohkem, siis ei tekkinud olukorda, kus see maht oleks ületatud. Nüüd aga, kui lihtsalt kõik PDF-failid kokku liita, juhtub tihti olukord, kus see maht ületatakse. Seega on vaja leida lahendus, kuidas vähendada PDF-ide mahtu, et need ei ületataks lubatud mahu piiri.

Soovitud mahu saavutamiseks sorteeritakse failid kahanevas järjekorras ja liidetakse kõikide mahtude summad kokku. Kui mahtude summa ületab viis protsenti väiksemat suurust kui 5 MB, siis hakatakse suuremate failide mahte vähendama, kuniks saavutatakse lubatud maht.

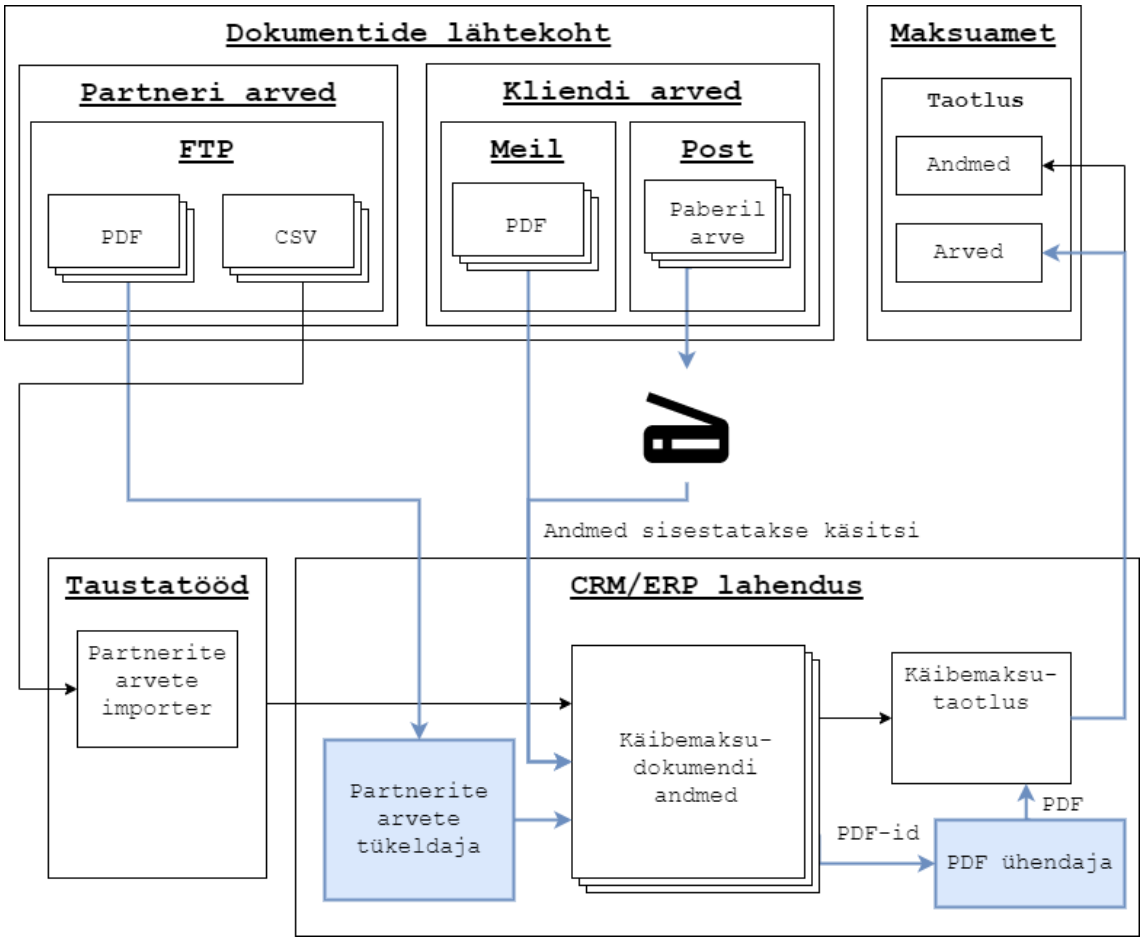
Autor uuris erinevaid variante PDF mahu vähendamiseks ja lõpuks langes valik GhostScript-i peale. See vähendas faili mahtu kõige rohkem sama kvaliteedikao juures. GhostScript võimaldab PostScript-i faili põhjal luua PDF-faili ja samuti on võimalik ka PDF-failist luua PostScript-i fail. Faili mahu vähendamiseks kasutatakse seda ära: konverteeritakse PDF GhostScript-i abil PostScript-i failiks ja seejärel tagasi PDF-failiks, kusjuures tagasi genereerimisel määratakse väiksemad graafilised nõuded. PDF-faili formaat on ehitatud PostScript-i peale [10, lk 23].

GhostScript-i interpretaatorist on olemas Windows-i keskkonnas DLL (*dynamic-link library*) [11]. Kuna tegemist ei ole .NET teegiga, ei saa seda otse kasutada .NET rakenduses. Seetõttu tuleb haldamata (*unmanaged*) DLL eraldi laadida rakenduse mällu ja pöörduda *interop*'i abil otse DLL-i poole [12]. Koodinäidis haldamata DLL-i kasutamisest on toodud lisas 3. Esmalt tuleb KERNEL32 abil laadida GhostScript-i DLL mällu. KERNEL32 on samuti haldamata DLL, seetõttu tuleb ka selle meetodid käsitsi paika panna. Edasi tuleb luua uus GhostScript-i interpretaatori protsess, meetod

gsapi_new_instance. Seejärel saab juba kutsuda välja konkreetset GhostScript API meetodit, joonisel meetod gsapi_init_with_args. Kõige lõpus tuleb ka kasutatavad DLL-id vabaks lasta, kuna muidu saab lõpuks mäluruum täis, mis on eraldatud haldamata DLL-idele.

4.1.4 Muudatused protsessis

Peale esimest iteratsiooni on käibemaksu tagastamise protsessis toimunud märkimisväärsed muudatused. Hetke protsess on ära toodud joonisel 3, muudatused on ära toodud sinise värviga. Kõige suurem muudatus protsessis on see, et enam ei ole vaja välja printida kõiki PDF-arveid. Partneri arvete PDF-id võetakse automaatselt FTP-st, tükeldatakse ära ja seostatakse konkreetse käibemaksudokumendiga. Kliendi arveid sisestatakse jätkuvalt käsitsi, kuid nüüd on tekitatud võimalus arve PDF kohe CRM/ERP lahenduses olevale käibemaksudokumendile külge panna. Kliendi poolt paberil saadetud arved tuleb sisse skaneerida ja käsitsi sisestada CRM/ERP lahendusse. Käibemaksutaotluse tegemisel võetakse nüüd lisaks käibemaksudokumentide põhiandmetele ka kõikide nende PDF-id ja liidetakse need kokku üheks koond-PDF-iks, mida varem tehti kõikide dokumentide koos skaneerimisel.



Joonis 3. Käibemaksu tagastamise protsess peale esimest etappi.

4.2 Integreeritud realisatsioon

Olemasolev CRM/ERP lahendus on kirjutatud .NET Framework raamistiku peale, kasutades C# programmeerimise keelt. Kuna lahendus tuleb integreerida olemasolevasse süsteemi, tuleb ka loodav lahendus luua kasutades .NET Framework raamistikku. Lahenduse juures kõige olulisem on PDF-ist kätte saada olulised andmed.

PDF-dokument on mõeldud andmete lõplikuks kuvamiseks, kus midagi enam ei muudeta. Sellest tulenevalt on tegemist graafilise,, mitte tekstilise formaadiga. Dokumendis on kirjas kogu informatsioon, mis on vajalik dokumendi kuvamiseks. Teksti kuvamiseks kasutatakse tekstielemente, mis määravad ära teksti asukoha, fondi ja suuruse. Tekstielement võib määrata ainult ühe märgi, terve sõna või kogu lõigu. Teksti kodeerimiseks ei kasutata standardset kodeeringut, nagu näiteks Unicode-i, vaid konkreetne kodeering on määratud PDF-dokumendis, fondi objektis. Fondi objekt kirjeldab ära, milline iga märk välja näeb ja mis on selle vastava märgi kodeering. Kuna

üks sõna võib koosneda mitmest tekstielemendist, ei ole võimalik lihtsalt lahtikodeerimisega teada saada, mis täpselt PDF-dokumendis kirjas on. Samuti ei pruugi olla tekstielementides kirjas tühikuid, vaid järgmise tekstielemendi asukoht on sobivas kohas, et kuvamisel tekiks tühik. Kõige selle tõttu oleks keeruline ja vägagi ajamahukas ise luua lahendus, mis oskaks PDF-dokumendist teksti loetaval kujul kätte saada. Seetõttu oleks mõistlik leida mingi sobiv teek, kus on see kõik juba ära tehtud [10, lk 404–432].

Autor katsetas erinevaid teeke ja lõpuks langes valik Apache PDFBox-i [13] peale. Tegemist on küll Java teegiga, kuid IKVM.NET [14] abil on võimalik Java teeke ja koodi kasutada .NET Framework raamistiku peal. IKVM.NET puhul on tegemist .NET Framework teegiga, mis implementeerib Javat. See teek sisaldab endas Java virtuaalmasina ja Java klasside implementatsiooni ning tööriistu, mis panevad omavahel suhtlema Java teegid ja .NET rakenduse.

4.2.1 PDF-arvetest andmete lugemine

Autor valis lähenemise, et iga erineva teenusepakkuja arve kohta luuakse eraldi koodiklass, mis oskaks konkreetset tüüpi arve pealt välja lugeda kõik olulised andmed. Edaspidi nimetatakse neid klasse *parser*'iteks Kuid selleks, et millegi alusel arve väljastanud teenusepakkujat tuvastada, tuleb esmalt kogu PDF-dokumendi peal olev tekst saada masinloetaval kujul kätte. Selleks kasutatakse Apache PDFBox-i teeki, mille abil teisendatakse PDF-dokument teksti kujule. Joonisel 4 on toodud välja kasutatava koodi näidis selle tegemiseks.

```
PDFTextStripper pdfTextStripper = new PDFTextStripper();
pdfTextStripper.setSortByPosition(true);
pdfTextStripper.setLineSeparator("\n");
string text = pdfTextStripper.getText(document).Replace("\r", "");
```

Joonis 4. PDF-arvelt kogu teksti kättesaamise koodinäidis.

Seejärel, kui arve tekst on kätte saadud, saab seda kasutada arve tüübi tuvastamiseks. Selleks kontrollitakse erinevate võtmesõnade/võtmefraaside sisaldumist arve tekstis. Nendeks on näiteks teenusepakkuja firma nimi, aadress, käibemaksukohuslase number, firma kodulehe veebiaadress. Kui tüüp on tuvastatud, kasutakse spetsiaalselt loodud *parser*'it dokumendist vajaliku informatsiooni kättesaamiseks. Üldine ülesehitus on kõikidel *parser*'itel sarnane. Esmalt loetakse tekstina sisse kogu PDF-dokument.

Seejärel kasutatakse regulaaravaldise (*regular expression*) lauseid, et kätte saada konkreetselt vajalik informatsioon.

Esmalt tuleb PDF-arve pealt kätte saada põhilised andmed. Selleks on loetud varasemalt kogu tekst PDF-arvelt välja. Joonisel 5 on toodud välja võrdlus, kuidas on tekst PDF-arvel ja kuidas see sama tekst on kätte saadud teksti kujul. Konkreetselt vajalike andmete kättesaamiseks luuakse regulaaravaldis iga olulise andmevälja jaoks. Joonisel 6 on välja toodud koodinäidis, kuidas saadakse kätte PDF-arve pealt kliendi nimi. Teised väljad on tehtud analoogselt, erinev on regulaaravaldise tekst. Summade ja kuupäevade puhul teisendatakse tekst sobivaks andmetüübiks.


```
VAT INVOICE
Invoice number FIN10000001
Date 28.02.2018
Period 01.02.2018 - 28.02.2018
Seller Neste Markkinointi Oy
Customer no 012345657
Customer NÄIDIS FIRMA OÜ
Tänav tn. 10
12345 Tallinn, Harjumaa
Eesti
VAT registration no EE123456789
Registration no 12345678
```

Joonis 5. Põhiandmed PDF-is (vasakul) ja teksti kujul (paremal).

```
var customerRegex = new Regex(
      @"(?<=\n)(Customer no).*\n(Customer) +(?<customer>.+?)(?=\n)");
var customerMatcher = customerRegex.Match(text);
if (customerMatcher.Success)
{
    customerName = customerMatcher.Groups["customer"].Value;
}
```

Joonis 6. Programmikoodi näide kliendi nime lugemiseks.

Käibemaksu tagastamise jaoks on oluline kätte saada arve peal olevad tooted. Selle jaoks tehti keerulisem regulaaravaldis, mis leiab ühekorraga kogu tooterea informatsiooni. Joonisel 7 on toodud näidis, kuidas on tooted PDF-arve peal ja samad tooted ka teksti kujul. Joonisel 8 on toodud näide programmikoodist, kuidas näeb välja loodud regulaaravaldis ja kuidas selle abil toodete info kätte saadakse.

PURCHASES IN FINLAND, EUR:

Product	Litres	Net price	Net sum	VAT %	VAT sum	Gross sum
DI-29/-34	502.25	1.0661	535.45	24	128.51	663.96
DI-15/-25	1366.85	1.0674	1459.01	24	350.16	1809.17
AdBlue	31.52	0.5508	17.36	24	4.17	21.53
TOTAL:	**1900.62**		**2011.82**		**482.84**	**2494.66**

(a)

```
PURCHASES IN FINLAND, EUR:
Net Net VAT VAT Gross
Product Litres price sum % sum sum
DI-29/-34 502.25 1.0661 535.45 24 128.51 663.96
DI-15/-25 1366.85 1.0674 1459.01 24 350.16 1809.17
AdBlue 31.52 0.5508 17.36 24 4.17 21.53
TOTAL: 1900.62 2011.82 482.84 2494.66
```

(b)

Joonis 7. Tooted PDF-arve peal (a) ja tooted teksti kujul (b).

```
var productRe = new Regex(@"(?<=\n)(?<name>.*?) +(?<quantity>\d+\.\d+)
*(\d+\.\d+) *(?<net>\d+\.\d+) *(\d+\.?\d*) *(?<vat>\d+\.\d+)
*(?<total>\d+\.\d+) *(?=\n)", RegexOptions.IgnoreCase);
var productMa = productRe.Matches(text);
var products = new ObservableCollection<ProductLine>();
foreach (Match match in productMa)
{
    products.Add(new ProductLine
    {
        IsChecked = false,
        Name = match.Groups["name"].Value,
        Net = decimal.Parse(match.Groups["net"].Value,
                    new NumberFormatInfo {NumberDecimalSeparator = "."}),
        Total = decimal.Parse(match.Groups["total"].Value,
                    new NumberFormatInfo {NumberDecimalSeparator = "."}),
        Vat = decimal.Parse(match.Groups["vat"].Value,
                    new NumberFormatInfo {NumberDecimalSeparator = "."})
    });
}
```

Joonis 8. Programmikoodi näide toodete info lugemiseks.

Peale toodete kättesaamist tuleb määrata arve peal olevad teenused. Seda tehakse arve peal olevate toodete nimede põhjal. Veel on oluline riik, kus arve väljastati, kuna olenevalt riigist, mille arve käibemaksu tagasi taotletakse, oleneb, milliseid teenuse koode kasutada tuleb. Kuna esialgu sai tehtud ainult paar teenusepakkujat ja erineva nimega tooteid ei olnud palju, otsustati, et vastavalt toote nimele tehakse *hardcoded*

lahendus, millega määratakse toote nimele vastavusse teenuse kood. Teenused saab jagada gruppidesse ja tihti on oluline just grupi esimene teenus, kuid riigiti võib see olla erinev. Seetõttu kasutati osaliselt lahendust, kus küsitakse andmebaasist selles riigis lubatud teenuste hulgast vastava grupi esimene teenus. Toote nimede vastavuse leidmiseks sai kasutatud jällegi regulaaravaldisi, kuna need võimaldab suuremat paindlikkust. Vahel võib toote nimi alata samamoodi, kuid lõpp võib olla veidi erinev. Sellisel juhul saab alguse põhjal juba paika panna, mis teenus sellele vastama peaks.

4.2.2 Valideerimine

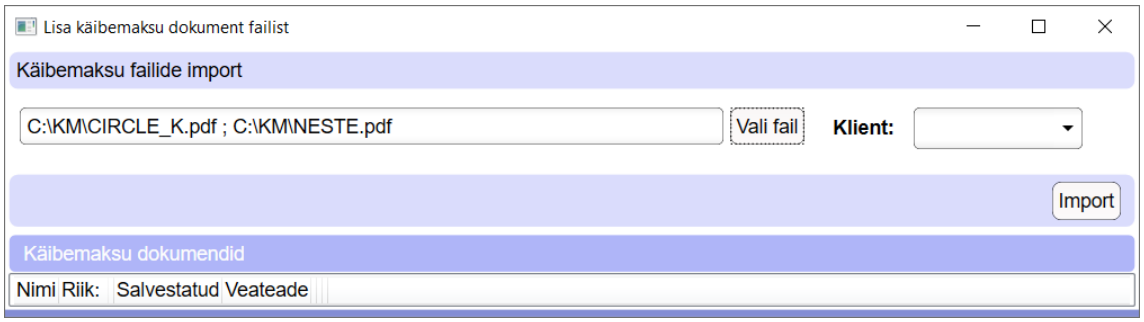
Enne kasutajale info kuvamist oleks mõistlik teha esmane dokumendist saadud andmete valideerimine. Antud protsessis kontrollitakse, kas kõik vajalikud andmed õnnestus dokumendist välja lugeda. Kui mitte, siis antakse teada, et antud dokumendist andmete saamisega oli probleeme. Teine kontroll on see, et vaadatakse, kas kõikide toodete summad klapivad arve kogusummadega. Vastavalt sellele kuvatakse juba kasutajaliideses veateade, kui valideerimisel tekkis vigu.

4.2.3 Kasutajaliides

Vastavalt äriloogika nõuetele tuli luua kasutajaliides, kus kohas toimuks uute dokumentide sisestamine. Seal peaks saama ka andmete õigsust kontrollida ja vajadusel andmeid parandada. Selleks loodi eraldi aken, mille sisuks on kõik automaatselt PDF-arvete sisestamisega seonduv. Kastajaliidese disainimisel on kasutatud sarnast joont kogu muu kasutajaliidesega.

Vastava akna avakuva on nähtav joonisel 9. Tekitatud on võimalus sisestada mitu faili korraga. Toetatud on failitüüpidest PDF, ZIP ja BDOC. Võimalus on valida ka klient, kelle arveid hakatakse sisestama. Sellest on kasu juhul, kui ei suudeta tuvastada dokumendist automaatselt klienti. Enamasti jäetakse see valik tühjaks ja klient tuvastatakse dokumendist automaatselt.

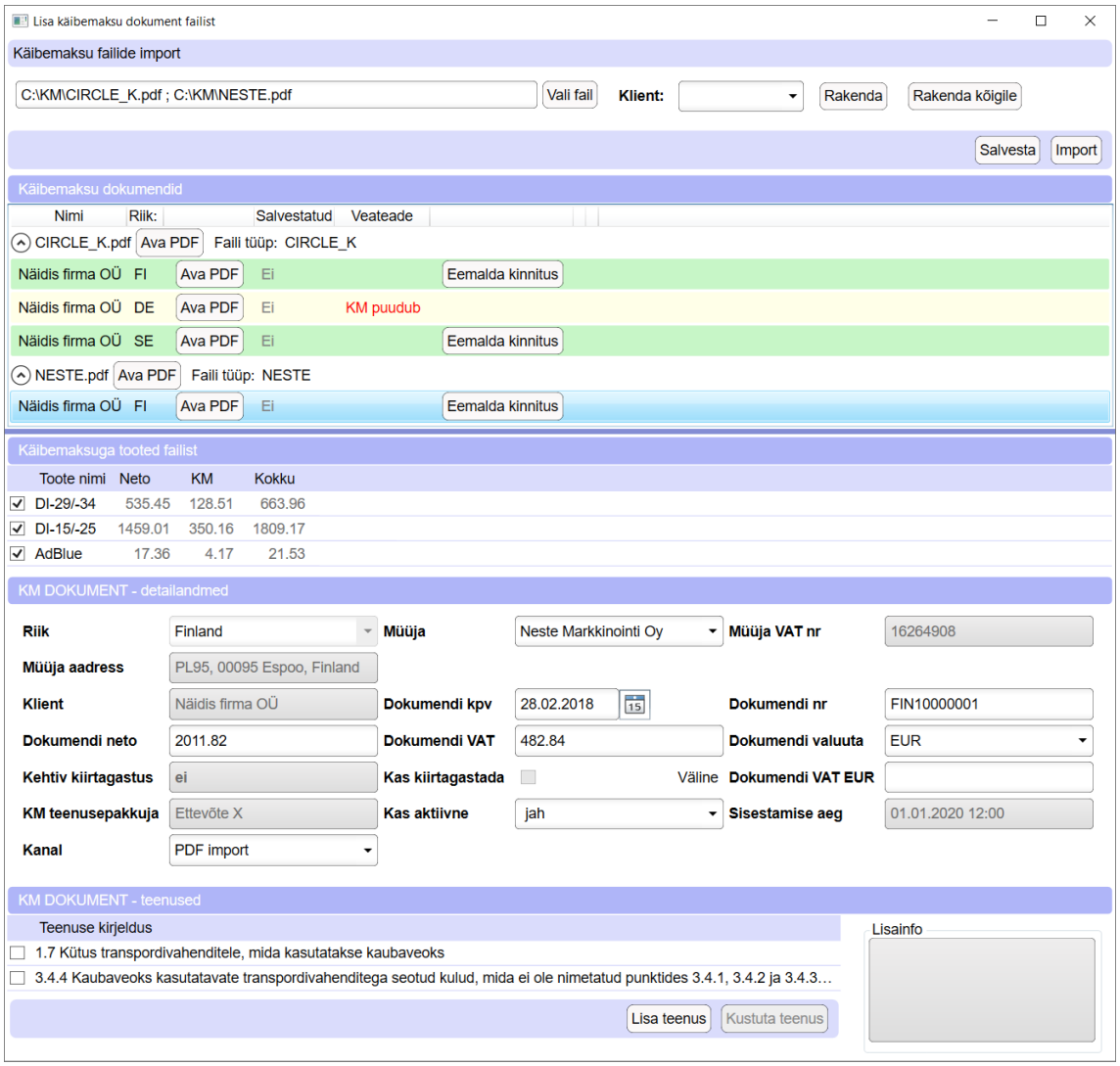
Automaatselt kliendi nime tuvastamiseks on mitmeid võimalusi. Üheks variandiks oleks seda teha nime põhjal, kuid erinevad ettevõtted võivad kliendi nime kirjutada veidi erinevalt. Kuna kõik ettevõtte X kliendid on juriidilised isikud, kellel on olemas käibemaksukohuslase number, siis esmajärjekorras kasutatakse seda kliendi automaatseks tuvastamiseks. Kui see ei õnnestu, proovitakse seda teha kliendi nime alusel. Kui ka see ei õnnestu, tuleb klient kasutajaliideses käsitsi paika panna.



Joonis 9. Käibemaksudokumendi sisestamise akna avakuva.

Peale PDF-arvete importimist avaneb kuva, mis on nähtav joonisel 10. Detailandmete ja teenuste osa on taaskasutatud juba olemasolevas CRM/ERP lahenduses olevast kasutajaliidesest. Töötajate jaoks on sedasi uut lahendust lihtsam kasutusele võtta, kui see on sarnane olemasoleva lahendusega. Samuti on arendusele kuluv aeg väiksem.

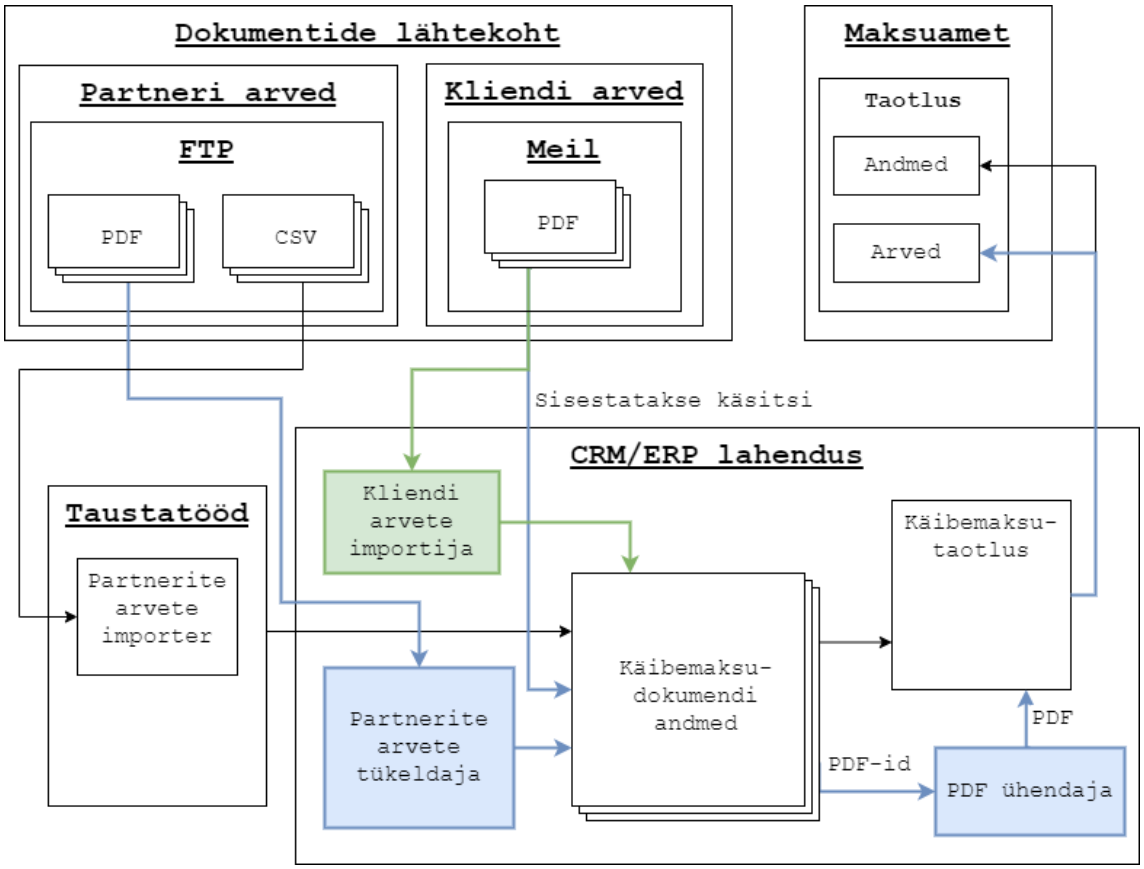
Kui kõik andmed on üle vaadatud ja andmete õigsuses on veendutud, saab „Salvesta“ nupu vajutamisega salvestada CRM/ERP lahendusse kõik kinnitatud käibemaksudokumendid. Kui on soov mõnda neist mitte salvestada, tuleb vastavatelt dokumentidelt enne salvestamist kinnitus eemaldada. Vastav lähenemine sai valitud, kuna enamasti soovitakse kõik imporditud PDF-dokumendid ka süsteemi salvestada. Kui kõigil oleks eraldi salvestusnupp, kuluks salvestamise peale rohkem aega ja tuleks teha lisanduvaid nupuvajutusi.



Joonis 10. Käibemaksudokumendi sisestamise aknakuva peale importimist.

4.2.4 Muudatused protsessis

Praeguseks hetkeks on ettevõtte X töötajate poolt tehtud klientidele piisavalt teavitustööd, et praktiliselt ei tule enam ühtegi arvet klientide poolt paberi peal. Kõik arved saadetakse meili teel. Selle etapi järgne protsessijoonis on ära toodud joonisel 11, muudatused on toodud rohelise värviga. Põhilise osa kliendilt saadud arvetest saab nüüd sisestada poolautomaatselt, kasutades loodud lahendust. Nende teenusepakkuja arveid, kellele ei ole veel *parser*'it loodud, saab sisestada käsitsi nagu varem. Selles arenduse etapis ei toimunud üldises protsessis rohkem muudatusi.



Joonis 11. Protsess peale partnerite arvete importija tegemist.

4.3 Teenused andmebaasi peale

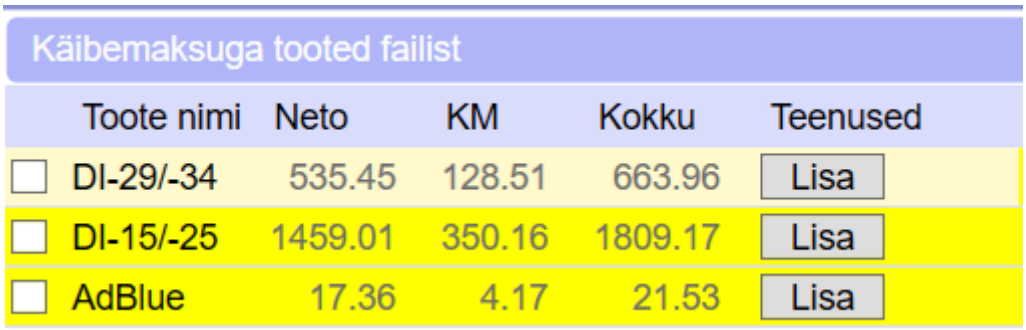
Esialgu sai tooted *hardcoded* lahendusega pandud vastavaks teenuseks. Antud lähenemine andis kiiremini tulemuse, kuna ettevõte X tegeleb logistikavaldkonnas, kus põhilised arved on teemaksud ja kütuse tasumised, mistõttu ei olnud arvete peal eriti palju erineva nimega tooteid. Kui maht läks suuremaks ja erinevaid teenusepakkujaid, kelle arveid automaatselt sisse lugeda, tuli juurde, siis hakkas järjest rohkem tekkima arvete peale tooteid, millele ei oldud paika pandud seost teenusega. Sellisel juhul pidi hakkama iga uue toote nime peale koodi muutma. See jällegi tõi kaasa, et kõik töötajad ettevõttes X pidid uue programmi alla laadima, mis tekitas jällegi ajakulu. Samuti tekitas see arendajate jaoks tööd juurde.

Esialgu nägi protsess välja selline, et ettevõtte X töötaja saatis arendajatele meili, kus mainiti ära, mis teenusepakkuja arvega tegemist on, milline on toote nimi ja mis kood sellele vastab. Arendaja pidid seejärel vastavad muudatused sisse viima. Seejärel tuli koodist jällegi programm valmis kompileerida ja edasi kõigi töötajate arvutisse uus

versioon peale panna. Versiooni vahetamine ettevõtte X töötajate jaoks on küll automaatne, kuid programmi avamisel peab ikkagi ootama senikaua, kuni programmi uuendatakse.

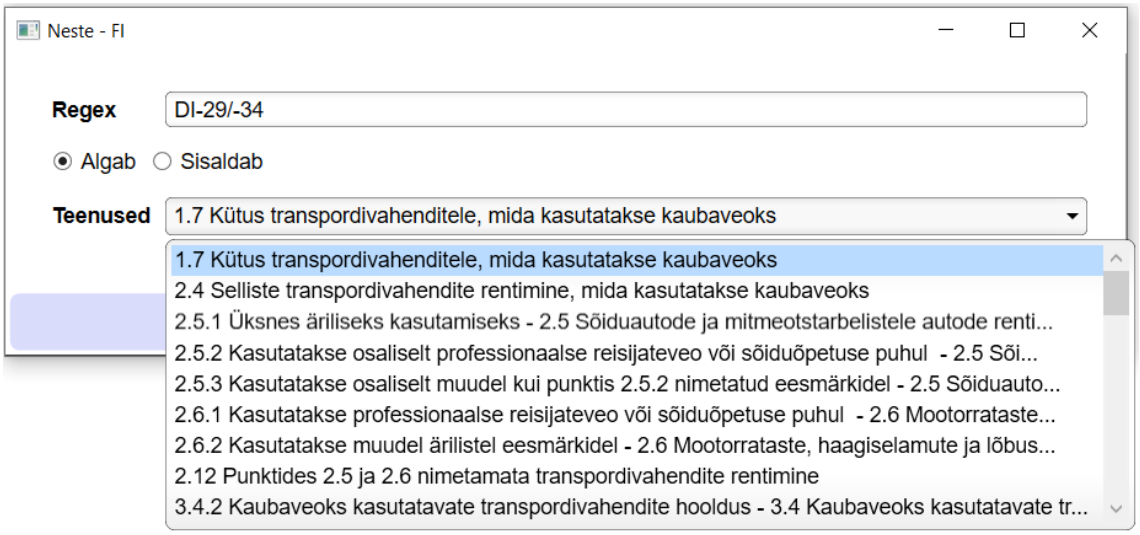
Kuigi tegutsetakse logistikavaldkonnas, tuli välja, et siiski on toodetel palju erinevaid nimekujusid: erinevate riikide arvetel erinevad tõlked. Seetõttu tuli leida lahendus, et iga uue toote nime tekkimisel ei peaks kogu programmi uuesti kompileerima ja välja vahetama. Lahenduseks sai viia toodete ja teenuste vaheline informatsioon andmebaasi peale.

Selleks sai lisatud kasutajaliidesesse toodete juurde eraldi veerg teenuste kohta. Igale tootele hakkab vastama konkreetne teenus. See on nähtav joonisel 12. Kui tootele vastavat teenust ei leita, saab nupuga „Lisa“ selle lihtsalt lisada.



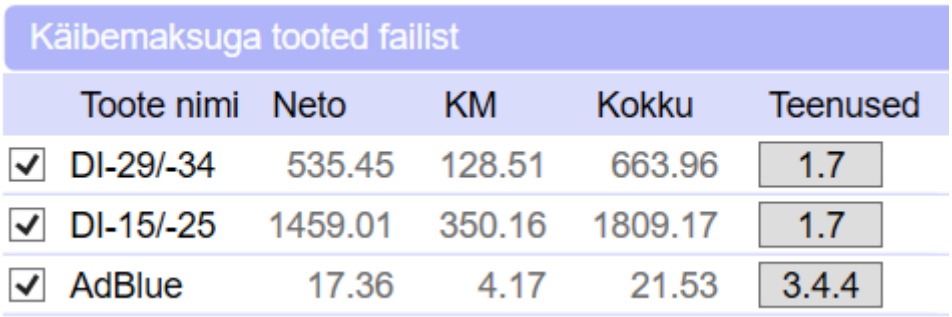
Joonis 12. Toodete info kasutajaliideses, kui antud tootele ei leitud vastavat teenust.

Lisamise nupu vajutamise peale avaneb aken, mis on nähtav joonisel 13. Rippmenüüs on nähtavad kõik selles riigis lubatud teenused käibemaksu tagastamisel, mille vahelt saab valida tootele vastava teenuse. Toote tuvastamiseks on kasutusel regulaaravaldis. Vaikimisi täidetakse see ära toote nimega. Selleks, et rakenduse kasutaja ei peaks ise käsitsi regulaaravaldist kirjutama, on lisatud võimalus valikuteks, kas algab selle sõnaga või sisaldab seda sõna. Algab puhul lisatakse regulaaravaldise algusesse „^“, mis tähendab, et toote nimi peab algama konkreetse sõnaga. Sisaldab puhul ei lisata midagi juurde, kuna vaikimisi otsib regulaaravaldis igalt poolt sõna/lause seest vastet.



Joonis 13. Kasutajaliides toodetele teenuse lisamiseks.

Kui tootele leiti teenus külge, siis kuvatakse lisamise nupu asemel teenuse number, nagu on nähtav joonisel 14. Kui peaks juhtuma, et tegu on vale teenusega, siis teenuse numbri peale vajutades avaneb sama aken, mis toote teenusega sidumisel. Seal on võimalik muuta regulaaravaldist, teenust või see reegel üldse ära kustutada.



Joonis 14. Toodete info kasutajaliideses, kui leiti automaatselt teenused külge.

Kokkuvõttes tegi teenuste andmebaasi peale viimine lihtsamaks nii arendajate kui ka ettevõtte X töötajate töö. Enamasti piisab teenuse lisamiseks ainult „Lisa“ nupu vajutamisest ja rippmenüüst sobiva teenuse vastavusse valimisega.

4.4 Mikroteenuseks realisatsioon

Varasemalt tehti monoliitseid lahendusi, kus kogu lahendus oli üks suur koodibaas. Kui oli vaja mingit osa rakendusest muuta, tuli kogu lahendus uuesti kompileerida ja välja vahetada. Mikroteenuste puhul on monoliitne rakendus löödud eraldi väiksemateks

tükkideks. Iga mikroteenus on eraldiseisev tükk rakendusest, mis teeb ühte kindlat asja. Igaüks neist on eraldi kasutusele võetav (*deployable*), erineval platvormil ja võib olla arendatud kasutades erinevat tehnoloogilist raamistikku (*technological stack*). Mikroteenus suhtleb teiste mikroteenustega kasutades API-t, enamasti RESTful (*representational state transfer*) API-t. Võrreldes monoliitse rakendusega on mikroteenuse eeliseks just tehnoloogiline vabadus. Saab kasutada uuemaid tehnoloogiad ja ei pea olema kinni juba kasutuses olevates tehnoloogiates. Lisaks on mikroteenuse kood lihtsamini hallatav. Koodi maht on tunduvalt väiksem kui monoliitsel rakendusel ja see suur koodi hulk on eraldi ära jagatud erinevate mikroteenuste vahel. Mikroteenuste arendamine toob endaga tihti kaasa CI/CD lähenemise ja rakenduse konteineriseerimise (*containerization*). Seda kõike selleks, et arendamise protsess, rakenduse paigaldamine ja väljavahetamine oleks kiirem ja lihtsam [15].

Esimese tõuke mikroteenuseks realisatsiooni juures andis asjaolu, et ühe teenusepakkuja arvetelt ei õnnestunud enam andmeid lugeda Apache PDFBox-i vanema versiooniga, kuid uuemal versioonil ei olnud IKVM.NET teeki sellest tehtud. Sai katsetatud uuema versiooniga ja seal toimis teksti lugemine. Katsetamiseks tegi autor lihtsa Java rakenduse, kuna originaalis on Apache PDFBox-i puhul tegemist Java teegiga. Küll oleks saanud selle ühe teenusepakkuja arvete puhul ka mingi teistsuguse lahenduse välja mõelda, kui see oleks olnud ainuke põhjus.

Teiseks põhjuseks võib tuua selle, et juba mõnda aega on plaan töölaua rakendus viia veebirakenduseks. Arendus sellega juba käib aktiivselt. Veebiteenuse puhul ei oleks mõistlik seda esirakenduses (*front-end*) teha ja seetõttu oleks mõistlik teha seda eraldi mikroteenuses. Uued serverid, kus teenused töötavad, on nüüd Linuxi masinad. Soov on Windowsi masinad kasutusest välja jätta. Seetõttu ei ole ka kasutatav olemasolev kood, mis on tehtud .NET Framework-i peale. Linuxi all ei ole sellele tuge. Variant ongi, kas teha nüüd see Javas, kus on kirjutatud kasutatav teek, või .NET maailmas on olemas .NET Core, mis toimib ka Linuxi peal. See ei toeta jällegi IKVM.NET-i ning seetõttu jääb ka lihtsalt *build target*'i muutmisest väheks.

Autor otsustas lõpuks ikkagi Java kasuks. Põhiargumendiks sai see, et Apache PDFBox puhul on ikkagi tegemist Java teegiga, ning kui teha juba mikroteenuseks, ei ole vajadust kasutada hetkel kasutuses olevaid tehnoloogilisi raamistikke.

Kuna olemasolev kood on IKVM.NET abil kirjutatud, siis põhiline osa koodist on ka juba sobiv Java kood. Suurt osa olemasolevast koodist saab kohe kasutusele võtta.

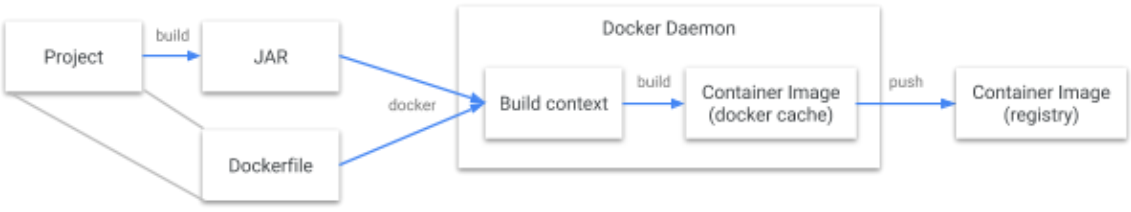
Programmeerimisekeelena sai kasutusele võetud Kotlin [16] ja raamistikuna Spring Boot [17]. Need võimaldavad lihtsat ja kiiret mikroteenuse arendamist.

4.4.1 Docker

Mikroteenuse lihtsamaks ülesseadmiseks otsustati, et kogu lahendusest luuakse Docker-i konteineri tõmmis. See võimaldab hiljem serveris lihtsalt üles seada mikroteenuse. Samuti on hilisem versioonivahetamine kergem. Docker-i konteiner sisaldab kõike, mida on programmi tööks vaja. Eraldi ei pea hakkama serverisse midagi muud paigaldama, ainuke nõutav programm on Docker daemon, mis tegeleb konteinerite tööle panemise ja töös hoidmisega [18].

Docker-i konteineri tõmmise loomiseks on kasutusel Jib. Selle puhul on tegemist avatud lähtekoodiga Java tööriistaga, mida haldab Google. Jib ehitab valmis optimeeritud Dockeri konteineri tõmmise (*container image*) [19].

Kui ei kasutaks Jib-i, tuleks Java rakendusest konteineri loomiseks esmalt ehitada Java lähtekoodist JAR fail. JAR faili puhul on tegemist Java rakenduse paketi failiformaadiga. See sisaldab kompileeritud Java lähtekoodi ja muid ressursse, millest rakendus koosneb. Standardsel juhul konteineri tõmmise ehitamine on toodud joonisel 15. Edasi tuleb luua Dockerfile. Selles failis määratakse ära, kuidas ehitatakse valmis konteineri tõmmis. Java rakenduse puhul tuleks kopeerida ehitatud JAR fail konteineri sisse ja määrata sisenemispunktiks (*entrypoint*) vastav käsk, mis paneb selle JAR faili tööle.



Joonis 15. Java rakendusest Dockeri konteineri tõmmise loomise protsess.

Docker-i konteineri tõmmis on üles ehitatud kihtide peale [18, lk 3]. Kõige aluseks võetaks mingi olemasolev tõmmis. Antud rakenduse puhul sobiks tõmmis, mis oskaks

jooksutada Java Spring Boot rakendust. Kui iga kord terve JAR fail panna konteinerisse, siis kihtidest saaks ära kasutada ainult baaskonteineri tõmmist. Programm aga sisaldab ka suures osas ressursse, mis ei muutu eriti tihti. Nendeks oleks kasutatavad teegid ja ressursifailid. Teegid muutuvad vaid siis, kui nende versiooni on aeg-ajalt vaja uuendada. Programmi lähtekood muutub palju sagedamini.

Antud juhul saaks kihtide loogikat ära kasutada, kui panna vähem muutuvad asjad varem konteineri tõmmisesse. Sellisel juhul hoitakse ruumi ja aega kokku, kuna ei pea muutumata kihte uuest looma. Lähtekoodi osa on mahu poolest väike ja sedasi saaks konteineri tõmmise ehitamisega kiiremini valmis ning võtaks ka vähem ruumi registris. Samuti oleks uue versiooni allalaadimine kiirem, kuna alla tuleb laadida ainult muutunud kihid.

Selleks ongi kasutusel Jib, mille puhul tehakse kõik see kasutaja poolt taustal ära. Võrdluseks on toodud Jib-i protsess joonisel 16. Selle puhul antakse Jib-ile ette projekt ja lõpetuseks on konteiner üles laaditud etteantud registrisse. Seda kõike annaks ka käsitsi teha, kuid sellisel juhul läheks kogu Java rakenduse pakkimise protsess keerulisemaks.

Joonis 16. Java rakendusest Dockeri konteineri loomise protsess Jib-i kaasabil.

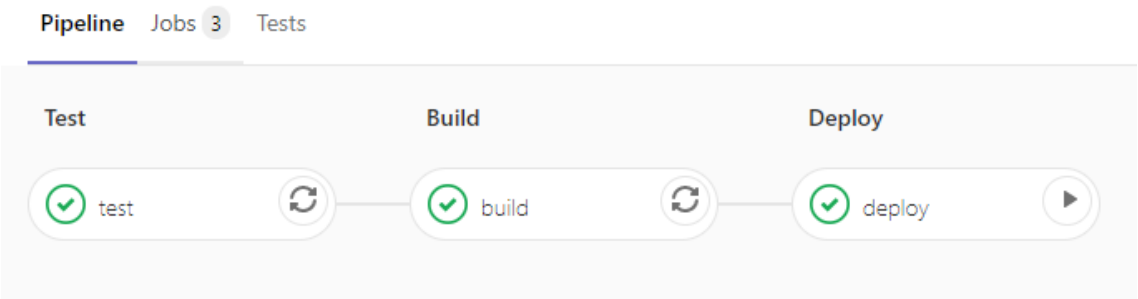
4.4.2 GitLab CI/CD

Projektis võeti kasutusele CI/CD lähenemine. See hõlmab endas kolme metoodikat. Esimene on pidev integratsioon (*continuous integration*). See tähendab, et iga koodi muudatuse peale, mis pannakse git-i ples, ehitatakse (*build*) programm valmis ja lastakse testid käima. Eesmärgiks on vähendada koodi arendamise käigus tekkivate vigade arvu [20].

Teiseks on pidev edastamine (*continuous delivery*). See hõlmab endas programmi valmisseadmist sellisel viisil, et kasutajapoolse käivitamise peale laaditakse uus versioon serverisse ülesse ja pannakse käima.

Kolmandaks on pidev juurutamine (*continuous deployment*). See on samm edasi pidevast edastamisest, kus serveris versiooni vahetamine tehakse ära automaatselt, ilma kasutajapoolse sekkumiseta.

Antud projektis kasutatakse esimest kahte metoodikat: pidev integratsioon ja pidev edastamine. Projekti lähtekoodi hoitakse GitLab-is, seetõttu võeti kasutusele GitLab CI/CD, mis on sisseehitatud juba GitLab-i. Selle kasutamiseks tuleb lisada repositooriumisse `.gitlab-ci.yml` fail, mis sisaldab käsklusi, mida käivitatakse iga kood üles laadimise peale git-i (*git push*). Antud projektis kasutatav `.gitlab-ci.yml` fail on ära toodud lisas 3. Selles failis on skriptid (*scripts*) grupeeritud eraldi töödeks (*jobs*) ning koos moodustavad need *pipeline*'i. Selles projektis on kolm tööd, projekti *pipeline*'i skeem GitLab-is on toodud joonisel 17.



Joonis 17. GitLab-i *pipeline*'i skeem.

Esimene töö on testimine (*test*). Selle käigus käivitatakse kõik loodud testid. Kui mõni test peaks ebaõnnestuma, katkestatakse *pipeline* ja järgmisi töid enam käima ei panda.

Teiseks on Docker-i konteiner tõmmise ehitamine (*build*). Selle käigus ehitatakse valmis Docker-i konteiner tõmmis ja pannakse see üles GitLab-i Docker-i registrisse.

Kolmandaks tööks on juurutamine (*deploy*). Selle käigus vahetatakse serveris töötav versioon uue vastu välja. Selle käivitamiseks peab kasutaja ise vajutama *pipeline*-i juures nuppu, mille peale seda tehakse. Seda saab teha ainult juhul, kui kõik testid lähevad läbi ja Docker-i konteineri tõmmise ehitamine õnnestub.

4.4.3 Testimine

Koodi arendamise seisukohalt on oluline roll testimisel. Kliendi poolt saadetud teenusepakkujate PDF-arved võivad ajas mingil määral muutuda. Võib tulla juurde teises keeles sama teenusepakkuja arveid, millelt peaks andmed kätte saama. Sellistel

juhtudel tuleb koodi muuta, kuid siis on oht, et sama teenusepakkuja teiste PDF-arvetelt andmete lugemisel võivad tekkida vead. Kuna kõigi eelmiste versioonide käsitsi läbi testimine on liiga ajamahukas, loodi selle jaoks automaattestid.

Testimiseks on kasutusel JUnit 5 raamistik [21]. Testid on üles ehitatud PDF-failidest info kättesaamise õigsuse peale. Testimiseks on määratud eraldi kataloog, kuhu saab panna PDF-faili ja JSON-faili paarid. PDF on vastav dokument, mida testida ja JSON-failis on kirjas, mida failist leidma peab. JSON-faili struktuur on toodud joonisel 18.

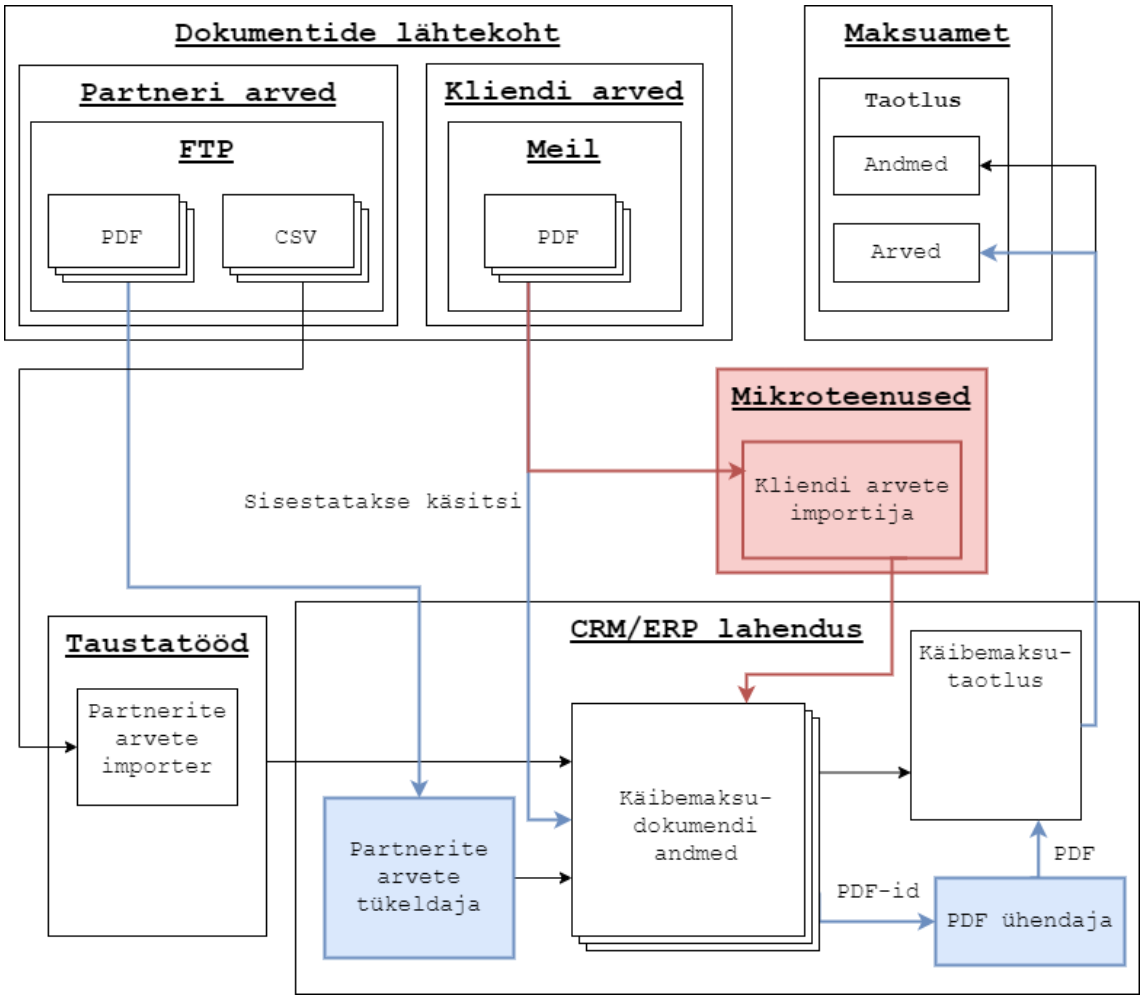
```
{
  "VatDocumentType": "NESTE"
  "CustomerName": [
    {
      "CustomerName": "NÄIDIS FIRMA OÜ",
      "CustomerVatPayerNo": "EE123456789",
      "DocumentNo": " FIN10000001",
      "DocumentDate": "2018-02-28",
      "SellerVatPayerNo": "FI16264908",
      "CountryCode": "FI",
      "DocumentCurrency": "EUR",
      "DocumentNet": 2011.82,
      "DocumentVat": 482.84,
      "Products": [ {
          "Name": "DI-29/-34",
          "Net": 535.45,
          "Vat": 128.51
        }, {
          "Name": "DI-15/-25",
          "Net": 1459.01,
          "Vat": 350.16
        }, {
          "Name": "AdBlue",
          "Net": 17.36,
          "Vat": 4.17
        } ]
    }
  ]
}
```

Joonis 18. Testimise JSON-faili struktuur.

4.4.4 Muudatused protsessis

Peale mikroteenuseks realisatsiooni ei toimunud ettevõtte X töötajate jaoks protsessis ühtegi muudatust. Protsess ise on toodud joonisel 19, muudatused on toodud punase

värviga. Taustal on viidud kliendi arvete importija eraldi mikroteenuse peale. Kasutajaliides jäi kõik samaks, kasutajatel tuleb ikkagi sisestada PDF-arved samast kohast. Taustal saadetakse see PDF mikroteenusele, mis seejärel tagastab CRM/ERP lahendusele dokumendis olevad andmed.

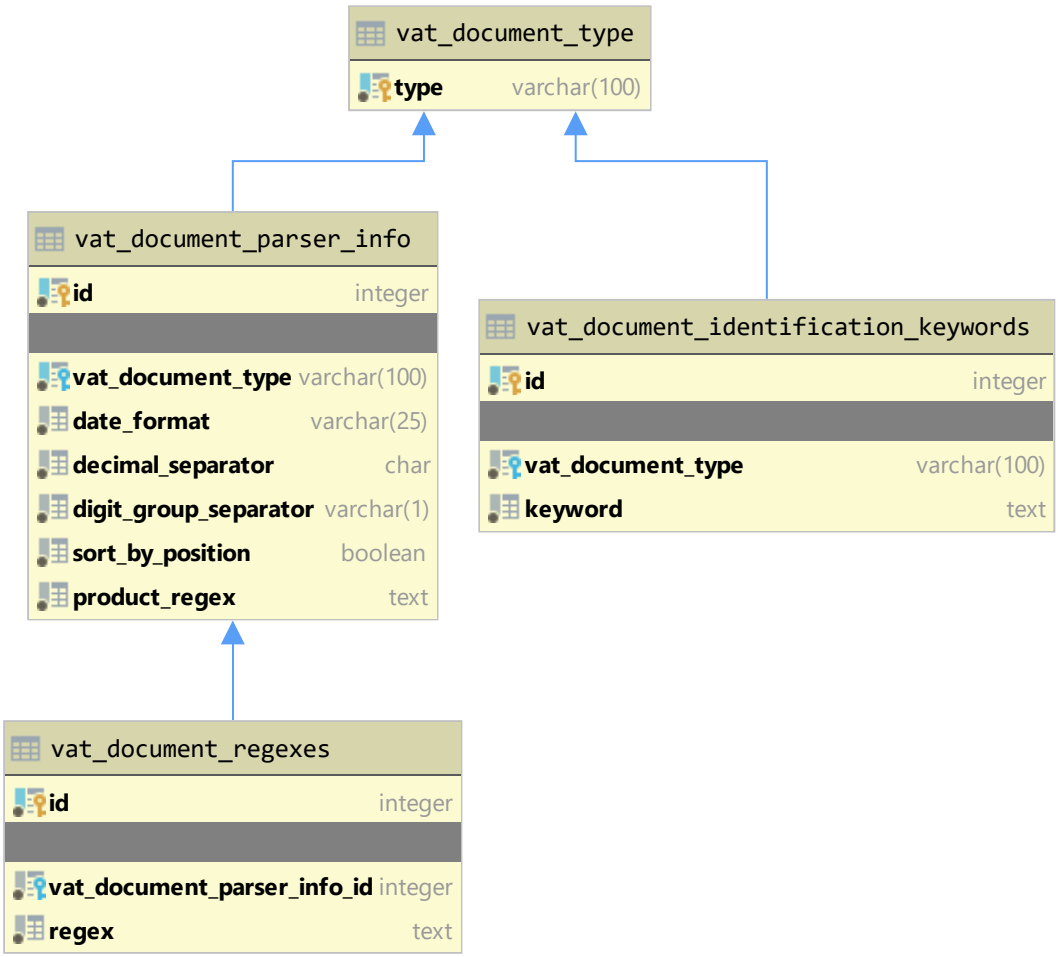


Joonis 19. Käibemaksu taotlemise protsess peale mikroteenuseks realisatsiooni.

4.5 *Parser*'i info andmebaasi peale

Kuna suurema osa *parser*'ite ülesehitus on sarnane, põhiliselt erinevad ainult regulaaravaldised ja olenevalt teenusepakkujast ka kuupäeva formaat, siis otsustas autor, et *parser*'ite muutuva osa saaks viia andmebaasi peale ja luua universaalsem *parser*, mis selle informatsiooni põhjal oskaks PDF-arvetelt olulised andmed välja lugeda.

Koos sellega tuleb PDF-arve tüübi tuvastamine viia andmebaasi peale. Kuna vastasel juhul tuleb uue teenusepakkuja lisandumisel koodi muuta. Selleks sai loodud mikroteenuse juurde eraldi andmebaas, millel puudub seos CRM/ERP lahenduse juures kasutatava andmebaasiga. Antud andmebaasi struktuur on toodud joonisel 20.



Joonis 20. Mikroteenuse andmebaasi struktuur.

PDF-arve tüübi tuvastamiseks loodi tabel vat_document_identification_keywords, kus määratakse ära erinevad märksõnad, mis peaksid eksisteerima konkreetse teenusepakkuja arve peal. Tüübi tuvastamiseks kasutatakse SQL lauset, mis on nähtav joonisel 21. Selleks vaadatakse, millisele arve tüübile vastavaid märksõnu on protsentuaalselt kõige rohkem PDF-arve tekstis.

Peale tüübi tuvastamist vaadatakse, kas vastava *parser*'i informatsioon on andmebaasis või koodis. Jäeti alles võimalus koodis *parser* valmis kirjutada, kuna osadel juhtudel võib olla tegemist keerulisema arvega, mida ei oleks võimalik lugeda üldise *parser*'iga.

Alati oleks võimalik tekitada võimalus lisada erandeid, kuid sellisel juhul läheks see juba liiga keeruliseks ja kaotaks enda esialgse mõtte. Üheks selliseks erandiks on näiteks Circle K arve, mida tuleb enne andmete lugemist tükeldada. Lisaks on olemas juba mingi hulk *parser*'eid, mis on realiseeritud koodis. Nende kõigi andmebaasi peale viimine tekitaks juurde lisatööd.

```
WITH matching_types AS (
    SELECT ide.vat_document_type
         , count(ide.vat_document_type)                    as matching_count
         , (SELECT count(1)
            FROM vat_document_identification_keywords
            WHERE vat_document_type = ide.vat_document_type) as total_count
    FROM vat_document_identification_keywords ide
    WHERE lower('PDF-ARVE TEKST') like '%' || keyword || '%'
    GROUP BY vat_document_type)
SELECT vat_document_type
FROM matching_types
ORDER BY matching_count * 100.0 / total_count DESC
LIMIT 1;
```

Joonis 21. SQL lause arve tüübi tuvastamiseks.

Parser'i info hoidmiseks loodi eraldi tabel vat_document_parser_info ja sinna juurde kuuluv tabel vat_document_regexes. Esimeses tabelis on kirjas üldine informatsioon *parser*'i kohta. Näiteks millisel kujul on arvel kuupäev ja mida kasutatakse numbrites kümnendkohtade eraldajana. Teises tabelis, vat_document_regexes, on kõik regulaaravaldise laused, mille abil leitakse üles olulised andmed arve pealt. Toodete leidmise regulaaravaldise lause on jäetud esimesse tabelisse, kuna selle abil tuleb leida üles kõik vastavused. Teiste puhul on üks konkreetne vaste, näiteks arve sisaldab ainult ühte arvenumbrit.

Regulaaravaldises saab otsingu tulemusi gruppidesse jagada ja neile gruppidele on võimalik nimesid anda [22]. Seda ära kasutades on paika pandud kindlad nimed, mille järgi otsustatakse, mida selle abil leitakse. Joonisel 22 on toodud regulaaravaldiste näiteid, mille abil leitakse üles valuuta (currency), dokumendi number (documentNo) ja dokumendi kuupäev (dockumentDate). Programmikoodi näidis, kus need kätte saadakse, on toodud joonisel 23. Kättesaadud väärtused pannakse sõnastiku (*dictionary*) andmetüüpi. Kõige lõpus korjatakse need sealt välja, teisendatakse teksti kujult õigele andmetüübile ja pannakse objekti klassis õige muutuja külge, mis seejärel tagastatakse CRM/ERP lahendusele.

```
(?<=\n)(PURCHASES IN|PIRKIMAI) [A-Z]+?, (?<currency>.+?):(?=\n)
(?<=\n)(Invoice number|Serija ir numeris) *(?<documentNo>.+?)(?= *\n)
(?<=\n)Dat(e|a) *(?<documentDate>\d\d\.\d\d\.\d\d\d\d)(?= *\n)
```

Joonis 22. Andmebaasis olevate regulaaravaldiste näiteid.

```
override fun initialize(text: String?) {
    super.initialize(text)

    for (regexLine in regexStringList) {
        val m = RegexHelper.getMatcher(regexLine.regexString, text)
        if (m.find()) {
            for (field in regexLine.fields) {
                fields[field] = m.group(field)
            }
        }
    }
}
```

Joonis 23. Koodinäide regulaaravaldiste põhjal andmete lugemiseks.

5 Tulemuste analüüs

Antud lõputöö raames valmis tarkvara, mis lihtsustab ettevõttes X arvete faktoorimise ja käibemaksu tagastamise protsesse. Antud projekti arendamine sai alguse juunikuus 2018. aastal ja arendamine on toimunud vahelduva eduga maikuuni 2020. aastal. Lõputöös toodud arendusetapid ei ole toimunud kõik kohe üksteise järel, vaid nende vahel on olnud kaks kuni kuus kuud pausi, mille jooksul kasutasid ettevõtte X töötajad senimaani loodud lahendust. Iga järgneva etapiga tehti ära järgmine toimiv minimaalne tükk, mida saaks kasutusele võtta. Algne plaan oli olemas, kuhu tahetakse lõpuks välja jõuda, kuid lahenduse kasutajatelt saadud tagasisidest ja autori poolt jooksvalt omandatud teadmistest tulenevalt korrigeeriti arenduse etappe jooksvalt. Lisaks lõputöö autorile on antud lahenduse juures töötanud veel üks arendaja, kelle ülesandeks on olnud kirjutada *parser*'eid erinevate teenusepakkujate arvetelt andmete lugemise jaoks.

5.1 Valideerimine

Analüüsi käigus, 3.4, pandi paika kindlad nõuded, millele loodud lahendus vastama peab. Üheks esimeseks nõudeks oli, et loodavale lahendusele peab saama ette anda mitu faili korraga ning need failid võivad olla ka kokkupakitud failiformaadi sees. See lahendati kasutajaliidese tasandil, kus saab ette anda mitu faili korraga. Seejärel edastatakse need failid ükshaaval mikroteenusele. Kokkupakitud failiformaadis andmed pakitakse lahti ja seejärel edastatakse sealsed PDF-arved mikroteenusele.

Järgmiseks nõudeks oli, et lahendus peab tuvastama arve tüübi automaatselt. Seda tehakse andmebaasi tasandil tekstist märksõnade otsimise teel. Edasi oli veel nõue, et enne CRM/ERP lahendusse salvestamist tuleb kuvada eelvaade ja anda võimalus vajadusel andmeid parandada. Selleks loodi CRM/ERP lahenduse juurde eraldi kasutajaliides, mis seda kõike võimaldab.

Viimaseks nõudeks oli, et loodab lahendus peab töötama nii käibemaksudokumentide, kui ka faktooritavate arvete puhul. See on ära lahendatud sellega, et suures osas on programmikood kirjutatud universaalsena. Vastavalt sellele, kas tegemist on käibemaksudokumendiga või faktooritava arvega, on osa koodi selle spetsiifiline.

Mõlema lahenduse puhul kasutatakse *parser*'eid, mille ülesehitus on suures osas sarnane, erinev on ainult see, mis andmeid sealt välja loetakse ja hilisem valideerimine. Kasutajaliides on ka mõlemal juhul sama. Faktooringuarvete jaoks on ära vahetatud ainult käibemaksudokumentide detailvaade faktooringuarvete detailvaatega.

Korrektse töötamise kindlustamiseks on loodud testid. Samuti tuleb tagasisidet ettevõtte X töötajatelt, kuna nad kontrollivad käsitsi arve andmed üle. Neilt on tulnud tagasi minimaalselt tagasisidet vigade kohta, mida rohkem tuli alguses. Viimasel ajal on hakatud loodud lahendust rohkem usaldama ja kui programm ütleb, et arvega on kõik korras, siis see kinnitatakse ilma üle vaatamata.

5.2 Äriline kasu

Ärilise poole pealt on tehtud lahendusest palju kasu. Ettevõtte X töötajate poolt on tulnud palju positiivset tagasisidet loodud lahenduse kohta. Antud lahenduse kasutuselevõtmine tegi võimalikuks kaugtöö, mis oli varem paberihunnikute tõttu mõeldamatu.

Kui enne antud lahenduse loomist tegeles käibemaksutaotluste tegemisega täistööajale taandatult 3,5 inimest, siis hetkel saavad sellega hakkama täistööajale taandatult 1,5 inimest. Samas mahud, millega tegeletakse, on kasvanud mitmekordseks. Lisaks on paberi kokkuhoid ligikaudu 30 000 lehekülge igas kalendrikuus, mis kõik prinditi varasemalt välja ettevõttes X. Need numbrid on ainult ühe riigi põhjal, kuna teistes riikides ei fikseeritud algushetke numbreid. Teistes riikides on need numbrid mõnevõrra väiksemad, kuid samuti on toimunud arvestatav kokkuhoid.

5.3 Eneseanalüüs

Lõputöö autorile andis antud lahenduse loomine hulgaliselt uusi teadmisi ja kogemusi. Mikroteenuste arendamisega, Docker-iga ja CI/CD-ga puudus autoril varasem kokkupuude.

Loodud lahendusega on autor üldjuhul rahul. Kui uuesti sama projekti teeks siis muudaks mõnda asja. Esialgne lahendus, kus otse .NET Framework raamistiku sees sai kasutatud Java koodi tekitas mõningaid probleeme. Kuna IKVM.NET teek sisaldab implementatsiooni kõikidest Java baasklassidest, hakkas IDE (*integrated development*

environment) pakkuma tihti Java tüüpe, mida mujal ei ole soov kasutada. Selle lahenduseks oleks olnud kirjutada IKVM.NET ja Apache PDFBox teeki sisaldav kood eraldi lahendusena (*solution*) projekti alla ja originaalis seda kasutatav lahendus oleks saanud viidata sellele teisele lahendusele ja kasutada seal valmis kirjutatud meetodeid. Sellisel juhul pakuks IDE ainult selles ühes lahenduses neid Java tüüpe ja meetodeid.

Lisaks oleks võinud kohe sisse tuua testimise. Päris kiiresti tekkis olukord, kus tuli *parser*'it muuta, kuna mõnelt sama teenusepakkuja arvelt ei osanud enam andmeid lugeda. Peale muudatuste tegemist tuli käsitsi üle kontrollida, kas varasemaid arveid oskab ikka sisse lugeda. Kui oleks testid olemas olnud, oleks see kontroll automaatselt ära tehtud ja oleks andnud ajalise kokkuhoiu. Samuti vähendaks see tekkivate vigade hulka.

5.4 Tulevikulahendusi

Loodud lahendus toimib ja on ettevõttes X muutnud kogu protsessi lihtsamaks ja kiiremaks, kuid leidub veel võimalusi tehtud lahendust täiendada.

Kuna süsteemi on sisestatud palju dokumente, millele on nüüd külge pandud PDF-arve, oleks need väga heaks sisendiks masinõppe jaoks. Masinõpet saaks kasutada kahel erineval otstarbel. Esimene lähenemine oleks luua masinõpet kasutades uusi *parser*'eid. Põhiline osa oleks just regulaaravaldiste loomine.

Teine lähenemine oleks luua universaalsem lahendus, mis oskaks arvetelt, ilma eelneva *parser*'i loomiseta, lugeda välja arve põhiandmed. Kuna iga arve sisaldab teatud informatsiooni, nagu näiteks arve number, arve kuupäev ja arve summad, siis need võiksid olla leitavad automaatselt üles, olenemata, mis teenusepakkuja arvega tegemist on. Selleks tuleb olemasolevaid andmeid ära kasutades luua piisavalt tark õppimisvõimeline süsteem.

6 Kokkuvõte

Lõputöö eesmärgiks oli luua ettevõttele X lahendus, mis aitaks lihtsustada käibemaksu tagastus taotluste tegemisprotsessi. Põhiline probleem oli PDF-arvete seostamine juba olemasolevate käibemaksudokumentidega ja PDF-arvetelt oluliste andmete välja lugemine. Lahenduse loomine algas esmalt olemasoleva protsessi analüüsiga, et viia ennast kurssi, mida tuleb asendama hakata. Seejärel uuriti juba turul olevaid lahendusi ja leiti neist ideid, mida enda lahenduses ära kasutada. Arendamise protsessi aluseks võeti MVP lähenemine.

Lõputöö käigus loodud lahendus arendati valmis mitme iteratsioonina, millest iga juures sai ettevõte X konkreetse võidu. Esmalt loodi partnerite PDF-arvete automaatne seostamine juba varasemalt CRM/ERP lahendusse sisestatud käibemaksu-dokumentidega. Seejärel loodi lahendus, mis aitaks klientide poolt saadetud arvetelt lugeda automaatselt olulisi andmeid ja nende põhjal teha CRM/ERP lahendusse valmis käibemaksudokumendid. Selle alusel loodi ka faktoorimist vajavatelt arvetelt automaatne lugemisvõimekus. Viimase etapina viidi loodud lahendus CRM/ERP lahendusest eraldi mikroteenuseks.

Kasutatud kirjandus

[1] V. Krusenvald, Faktooring ehk arvete rahastamine: mis see on & kuidas see töötab?, *Financer.com Eesti*, 17. jaanuar, 2020. [Online] https://financer.com/ee/laenud/arilaen/faktooring/ (25.04.2020).

[2] Piiriülene käibemaksu tagastamise süsteem – Maksu- ja Tolliamet [Online] https://www.emta.ee/sites/default/files/ariklient/tulu-kulu-kaive-kasum/kaibemaks/vat_juhend.pdf (25.04.2020).

[3] Nõukogu direktiiv 2008/9/EÜ, 12. veebruar 2008 [Online] https://eur-lex.europa.eu/legal-content/ET/TXT/PDF/?uri=CELEX:02008L0009-20101001&from=EN (25.04.2020).

[4] CostPocket (Tsekk.ee). [WWW] https://costpocket.com/et/ (02.05.2020).

[5] CostPocket OÜ. [WWW] https://www.funderbeam.com/company/wfinance-ou (02.05.2020).

[6] Docsumo | Automate Data Entry & Document Data Extraction. [WWW] https://docsumo.com/ (02.05.2020).

[7] *Docsumo Demo*. [Video fail] https://www.youtube.com/watch?v=OmTldphEM40 (02.05.2020).

[8] Docparser - Document Parser Software - Extract Data From PDF to Excel, JSON and Webhooks. [WWW] https://docparser.com/ (10.05.2020).

[9] A. Vartan ja J. Brinkerhoff, MVP: A maximally misunderstood idea, *Slalom*. [Online] https://www.slalom.com/insights/mvp-maximally-misunderstood-term (10.05.2020).

[10] PDF Reference – Adobe system incorporated. [Online] https://www.adobe.com/content/dam/acom/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf (10.05.2020).

[11] The Ghostscript Interpreter Application Programming Interface (API). [WWW] https://www.ghostscript.com/doc/9.52/API.htm (12.05.2020).

[12] DllImportAttribute Class (System.Runtime.InteropServices) [WWW] https://docs.microsoft.com/en-us/dotnet/api/system.runtime.interopservices.dllimportattribute (12.05.2020).

[13] Apache PDFBox | A Java PDF Library. [WWW] https://pdfbox.apache.org/ (15.05.2020).

[14] IKVM.NET Home Page. [WWW] https://www.ikvm.net/ (15.05.2020).

[15] A. Balalaie, A. Heydarnoori, ja P. Jamshidi, Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture, *IEEE Softw.*, kd 33, nr 3, lk 42–52, mai 2016, doi: 10.1109/MS.2016.64.

[16] Kotlin Programming Language, *Kotlin*. [WWW] https://kotlinlang.org/ (17.05.2020).

[17] Spring Boot. [WWW] https://spring.io/projects/spring-boot (17.05.2020).

[18] D. Jaramillo, D. V. Nguyen, ja R. Smart, Leveraging microservices architecture by using Docker technology, *SoutheastCon 2016*, märts 2016, lk 1–5, doi: 10.1109/SECON.2016.7506647.

[19] GoogleContainerTools/jib – GitHub. [WWW] https://github.com/GoogleContainerTools/jib (18.05.2020

[20] Introduction to CI/CD with GitLab | GitLab. [WWW] https://docs.gitlab.com/ee/ci/introduction/ (18.05.2020).

[21] JUnit 5. [WWW] https://junit.org/junit5/ (19.05.2020).

[22] Regex Tutorial - Named Capturing Groups - Backreference Names. [WWW] https://www.regular-expressions.info/named.html (22.05.2020).

Lisa 1 – Neste näidisarve PDF

NESTE

VAT INVOICE

Invoice number FIN10000001
Date 28.02.2018
Period 01.02.2018 - 28.02.2018

Seller **Neste Markkinointi Oy**
Customer no 01234567
Customer **NÄIDIS FIRMA OÜ**
Tänav tn. 10
12345 Tallinn, Harjumaa
Eesti
VAT registration no EE123456789
Registration no 12345678

PURCHASES IN FINLAND, EUR:

Product	Litres	Net price	Net sum	VAT %	VAT sum	Gross sum
DI-29/-34	502.25	1.0661	535.45	24	128.51	663.96
DI-15/-25	1366.85	1.0674	1459.01	24	350.16	1809.17
AdBlue	31.52	0.5508	17.36	24	4.17	21.53
TOTAL:	**1900.62**		**2011.82**		**482.84**	**2494.66**

SPECIFICATION

Card	Purchase date, time	Station	Receipt No	Product	Litres	VAT %	Gross sum
1234567890000015	01.02.2018 20:16	FI NESTE TRUCK NLI LUOLALA M	5001	DI-15/-25	271.20	24	359.85
1234567890000015	01.02.2018 20:22	FI NESTE TRUCK NLI LUOLALA M	5002	AdBlue	31.52	24	21.53
1234567890000023	01.02.2018 20:07	FI NESTE TRUCK NLI LUOLALA M	5003	DI-15/-25	500.04	24	663.49
1234567890000023	01.02.2018 20:16	FI NESTE TRUCK NLI LUOLALA M	5004	DI-15/-25	15.06	24	19.98
1234567890000023	23.02.2018 21:56	FI NESTE TRUCK NLI LUOLALA M	5005	DI-29/-34	502.25	24	663.96
1234567890000046	05.02.2018 13:06	FI NESTE TRUCK KEIMOLA ITA	5006	DI-15/-25	200.00	24	263.89
1234567890000046	07.02.2018 17:26	FI NESTE TRUCK MANTSALA	5007	DI-15/-25	250.54	24	330.48
1234567890000046	08.02.2018 07:06	FI NESTE TRUCK TKU VANHA TAMP	5008	DI-15/-25	130.01	24	171.48
			TOTAL:		**1900.62**		**2494.66**

Please note that only fuel purchases at Truck and Express stations are added to the invoice, due to legal restrictions.

Neste Markkinointi Oy
Keilaranta 21, 02150 Espoo, Finland
Domicile: Espoo

Registration no 1626490-8
VAT registration no FI16264908

Page: 1(1)

Lisa 2 – Circle K näidisarve PDF

NÄIDIS FIRMA OÜ

TÄNAV 10, TALLINN, TALLINN
EST - 12345 HARJUMAA

Euroopa ostude koond
(Ei ole arve)

Circle K Eesti AS

A.H.Tammsaare tee 47
EST - 11316 TALLINN

Tel:
Reg. Kood: 10180925
VAT Kood: EE100305317
Saldo teatis nr. 1234567890
Kuupäev: 30.09.2018
Kliendi number: 001234

Lk.: 1

Riik		Arve. nr	Valuuta	Summa käibemaksuga	Summa ilma käibemaksuta EUR	Summa käibemaksuga EUR
Finland	Arve	4321000242	EUR	350,49	282,65	350,49
Germany	Arve	4433243122	EUR	192,28	192,28	192,28
Sweden	Arve	1212320021	SEK	3.609,41	279,45	343,59
				KOKKU:	EUR	886,36

I N V O I C E

Customer:NÄIDIS FIRMA OÜ
Address: TÄNAV 10, TALLINN
EST - 12345 HARJUMAA
Country: Estonia
VAT No: EE123456789

Invoice No: 4321000242
Invoice Date: 30.09.2018
Customer No.: 123456 001234

Supplier: Circle K International Card Center AB
Address: Torkel Knutssonsgatan 24
S - 11888 STOCKHOLM
Country: Sweden
Vat No: FI18006785

Invoice for Purchases in: Finland

Supply Date	Voucher	Station-Town	Product/Service	Qty lit/pcs	Price/Unit	VoucherAm. incl.Vat	VoucherAm. excl.Vat	Disc. (-)	Surch. (+)	Miles	Net aft. Disc/Surch.	Vat %	Vat Amount	TOTAL	TOTAL
Supply Date	Voucher	Station-Town	Product/Service	Qty lit/pcs	Price/Unit	VoucherAm incl.Vat	VoucherAm. excl.Vat	Disc. (-)	Surch. (+)	Miles	Net aft. Disc/Surch.	Vat %	Vat Amount	TOTAL EUR	TOTAL EUR
Card Nr.: 0000000203			Card-Text: 123ABC KASUTAJ												
16.09.2018	0069	NESTE REVONLDI-10/-20		250,53	1,128	350,49	282,65	0,00	0,00	000000	282,65	24,00	67,84	350,49	350,49
										Total:	282,65		67,84	350,49	350,49
										TOTAL:	282,65		67,84	350,49	350,49

Product	Quantity	Net	Vat	Total
Diesel	250,53	282,65	67,84	350,49

Vat	Net	Vat	Total
24,00 %	282,65	67,84	350,49
Total (Currency:EUR)	282,65	67,84	350,49
Total (Currency:EUR)	282,65	67,84	350,49

R E C H N U N G

Kunde: NÄIDIS FIRMA OÜ
Adresse: TÄNAV 10, TALLINN
EST - 12345 HARJUMAA
Land: Estonia
Ust Nr: EE123456789

Rechnungs-Nr: 4433243122
Rechnungs-Datum: 31.09.2018
Kunden-Nr.: 123456 001234

Lieferant: Circle K International Card Center AB
Adresse: Torkel Knutssonsgatan 24
S - 11888 STOCKHOLM
Land: Sweden
UST-Nr: DE174497752

Rechnung für Bezüge in: Deutschland

Liefer-datum	Liefer-schein	Stations-Ort	Produkt/ Service	Menge L/kg/St	Preis/ Einheit	Bezugswert incl.Ust.	Bezugswert exkl.Ust.	Nachlaß exkl.U.	Aufschlag exkl.Ust	Kilo-meter	Netto-wert	Ust %	Ust Betrag	TOTAL	TOTAL
Supply Date	Voucher	Station-Town	Product/ Service	Qty lit/pcs	Price/ Unit	VoucherAm incl.Vat	VoucherAm. excl.Vat	Disc. (-)	Surch. (+)	Miles	Net aft. Disc/Surch.	Vat %	Vat Amount	TOTAL EUR	TOTAL EUR
Kartennr.: 0000000103			Kartentext: 345ABC KASUTAJ												
31.08.2018	00001797	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
01.09.2018	00001800	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
06.09.2018	00001815	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
08.09.2018	00001827	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
09.09.2018	00001828	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
10.09.2018	00001832	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
14.09.2018	00001844	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
15.09.2018	00001848	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
17.09.2018	00001855	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
19.09.2018	00001860	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
27.09.2018	00001890	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
28.09.2018	00001896	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
29.09.2018	00001898	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
										Summe:	108,68		0,00	108,68	108,68
Kartennr.: 0000000714			Kartentext: 456ABD KASUTAJ												
30.08.2018	00001793	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
06.09.2018	00001816	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
11.09.2018	00001837	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
13.09.2018	00001843	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
20.09.2018	00001862	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
21.09.2018	00001869	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
29.09.2018	00001900	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36

R E C H N U N G

Kunde: NÄIDIS FIRMA OÜ
Adresse: TÄNAV 10, TALLINN
EST - 12345 HARJUMAA
Land: Estonia
Ust Nr: EE123456789

Rechnungs-Nr: 4433243122
Rechnungs-Datum: 31.09.2018
Kunden-Nr.: 123456 001234

Lieferant: Circle K International Card Center AB
Adresse: Torkel Knutssonsgatan 24
S - 11888 STOCKHOLM
Land: Sweden
UST-Nr: DE174497752

Rechnung für Bezüge in: Deutschland

Liefer-datum / Supply Date	Liefer-schein / Voucher	Stations-Ort / Station-Town	Produkt/Service / Product/Service	Menge L/kg/St / Qty lit/pcs	Preis/Einheit / Price/Unit	Bezugswert incl.Ust. / VoucherAm incl.Vat	Bezugswert exkl.Ust. / VoucherAm. excl.Vat	Nachlaß exkl.U. / Disc. (-)	Aufschlag exkl.Ust / Surch. (+)	Kilometer / Miles	Nettowert / Net aft. Disc/Surch.	Ust % / Vat %	Ust Betrag / Vat Amount	TOTAL / TOTAL EUR	TOTAL / TOTAL EUR
										Summe:	58,52		0,00	58,52	58,52
Kartennr.: 0000000103			Kartentext: 123AST KASUTAJ												
18.09.2018	00001860	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
21.09.2018	00001867	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
27.09.2018	00001889	Internet	Eurovignett	1,00	8,000	8,00	8,00	0,00	0,36	000000	8,36			8,36	8,36
										Summe:	25,08		0,00	25,08	25,08
										TOTAL:	192,28		0,00	192,28	192,28

Produkte	Menge	Netto	Ust	Brutto
Others non site	23,00	192,28		192,28

Ust-Aufteilungs Übersicht	Netto	Ust	Brutto
%	192,28		192,28
Buchungsbetrag(Währung:EUR)	192,28	0,00	192,28
Buchungsbetrag(Währung:EUR)	192,28	0,00	192,28

F A K T U R A

Köpare: NÄIDIS FIRMA OÜ
Adress: TÄNAV 10, TALLINN
EST - 12345 HARJUMAA
Land: Estonia
Reg. nr.:EE123456789

Fakturanummer:1212320021
Fakturadatum: 31.09.2018
Kundnummer.: 123456 001234

Faktura för inköp i: Sverige

Säljare: Circle K International Card Center AB
Adress: Torkel Knutssonsgatan 24
S - 11888 STOCKHOLM
Land: Sweden
Reg. nr.: SE663000097101

Köp Datum	Kvitto nr.	Station Ort	Produkt/ Service	Volym lit/st	Pris/ volym	Brutto inkl.moms	Brutto utan moms	Rabatt (-)	Avgift (+)	Kilo meter	Netto eftr. Rab./Avgift	Moms %	Moms Belopp	Fakt.- belopp	Fakt. belopp
Supply Date	Voucher	Station- Town	Product/ Service	Qty lit/pcs	Price/ Unit	VoucherAm incl.Vat	VoucherAm. excl.Vat	Disc. (-)	Surch. (+)	Miles	Net aft. Disc/Surch.	Vat %	Vat Amount	TOTAL SEK	TOTAL EUR
Kort Nr.: 0000000103			Kort-Text:	123ABC KASUTAJ											
08.09.2018	00005431	Kristinehamn	Eurovignett	1,00	76,000	76,00	76,00	0,00	3,42	000000	79,42			79,42	7,72
20.09.2018	00001742	Balsta	Eurovignett	1,00	76,000	76,00	76,00	0,00	3,42	000000	79,42			79,42	7,64
26.09.2018	C00958	Luleå Persøn	ADBLUE	35,78	5,560	248,67	198,94	0,00	0,00	000000	198,94	25,00	49,73	248,67	23,66
26.09.2018	C00962	Luleå Persøn	MILES DIESE	200,03	12,968	3.242,49	2.593,99	96,01-	0,00	000000	2.497,98	25,00	624,50	3.122,48	297,02
											Totalt: 2.855,76		674,23	3.529,99	336,04
Kort Nr.: 0000000714			Kort-Text:	123DEF KASUTAJ											
27.09.2018	00012889	Graeddoe	Eurovignett	1,00	76,000	76,00	76,00	0,00	3,42	000000	79,42			79,42	7,55
											Totalt: 79,42		0,00	79,42	7,55
											TOTALT: 2.935,18		674,23	3.609,41	343,59

Produkt	Volym	Netto	Moms	Totalt
Diesel	200,03	2.497,98	624,50	3.122,48
Vehicle Acc.	35,78	198,94	49,73	248,67
Others non site	3,00	238,26		238,26

Moms		Netto	Moms	Totalt
	%	238,26		238,26
25,00	%	2.696,92	674,23	3.371,15

F A K T U R A

Köpare: NÄIDIS FIRMA OÜ
Adress: TÄNAV 10, TALLINN
EST - 12345 HARJUMAA
Land: Estonia
Reg. nr.:EE123456789

Fakturanummer:1212320021
Fakturadatum: 31.09.2018
Kundnummer.: 123456 001234

Säljare: Circle K International Card Center AB
Adress: Torkel Knutssonsgatan 24
S - 11888 STOCKHOLM
Land: Sweden
Reg. nr.: SE663000097101

Faktura för inköp i: Sverige

Köp Datum / Supply Date	Kvitto nr. / Voucher	Station Ort / Station-Town	Produkt/ Service	Volym lit/st / Qty lit/pcs	Pris/ volym / Price/ Unit	Brutto inkl.moms / VoucherAm incl.Vat	Brutto utan moms / VoucherAm. excl.Vat	Rabatt (-) / Disc. (-)	Avgift (+) / Surch. (+)	Kilo meter / Miles	Netto eftr. Rab./Avgift / Net aft. Disc/Surch.	Moms % / Vat %	Moms Belopp / Vat Amount	Fakt.-belopp / TOTAL SEK	Fakt. belopp / TOTAL EUR
Totalt	(Valuta: SEK)			2.935,18			674,23							3.609,41	
Totalt	(Valuta: EUR)			279,45			64,14								343,59

Lisa 3 – GhostScript DLL-i kasutamise koodinäidis

```
const string GSDLL32 = "gsdll32.dll";
const string KERNEL32 = "kernel32.dll";

[DllImport(KERNEL32)]
private static extern IntPtr LoadLibrary(string filename);

[DllImport(KERNEL32)]
private static extern bool FreeLibrary(IntPtr handle);

[DllImport(GSDLL32, EntryPoint = "gsapi_new_instance",
    CharSet = CharSet.Ansi, CallingConvention = CallingConvention.StdCall)]
private static extern int gsapi_new_instance(ref IntPtr pInstance,
    out IntPtr pCaller);

[DllImport(GSDLL32, EntryPoint = "gsapi_delete_instance",
    CharSet = CharSet.Ansi)]
private static extern int gsapi_delete_instance(IntPtr pInstance);

[DllImport(GSDLL32, EntryPoint = "gsapi_exit",
    CharSet = CharSet.Ansi)]
private static extern int gsapi_exit(IntPtr pInstance);

[DllImport(GSDLL32, EntryPoint = "gsapi_init_with_args",
    CharSet = CharSet.Ansi)]
private static extern int gsapi_init_with_args(IntPtr pInstance,
    int argc, [In, Out] String[] argv);

private IntPtr _handle;

public GhostScript()
{
    _handle = LoadLibrary(GSDLL32);
}

public long ReducePdfSize(string inputPath, string outputPath)
{
    string[] commands =
    {
        "reduce",
        "-dBATCH", "-dNOPAUSE", "-dQUIET",
        "-sDEVICE=pdfwrite", "-dCompatibilityLevel=1.4",
        "-dPDFSETTINGS=/printer",
        $@"-sOutputFile=""{outputPath}""", $@"""{inputPath}"""
    };
```

```
    CallGSDll(commands);
    if (File.Exists(outputPath))
        {
            FileInfo fi = new FileInfo(outputPath);
            return fi.Length;
        }
        return -1;
}

public void CallGSDll(string[] commandParameters)
{
    IntPtr ghostScriptPtr = IntPtr.Zero;
    gsapi_new_instance(ref ghostScriptPtr, out _);
    gsapi_init_with_args(ghostScriptPtr, commandParameters.Length,
commandParameters);
    gsapi_exit(ghostScriptPtr);
    gsapi_delete_instance(ghostScriptPtr);
    GC.Collect();
}

public void Dispose()
{
    if (_handle != IntPtr.Zero)
    {
        FreeLibrary(_handle);
        _handle = IntPtr.Zero;
    }

    GC.SuppressFinalize(this);
}
```

Lisa 4 – .gitlab-ci.yml faili näidis

```
stages:
  - test
  - build
  - deploy

variables:
  DOCKER_DRIVER: overlay2
  SPRING_PROFILES_ACTIVE: gitlab-ci

test:
  image: gradle:6.0.1-jdk13
  stage: test
  script: gradle test
  tags:
  - docker
  - pdf_reader_service-tests

build:
  image: gradle:6.0.1-jdk13
  stage: build
  script: gradle jib
  tags:
  - docker

deploy:
  image: deploy:prs
  script:
  - ssh ${ENV_USER}@${ENV_IP} "cd ${ENV_DOCKER_COMPOSE_DIR} &&
    docker login ${CI_REGISTRY} \
    -u ${CI_REGISTRY_USER} -p ${CI_REGISTRY_PASSWORD} &&
    docker pull ${CI_REGISTRY_IMAGE}:${CI_COMMIT_SHA} &&
    docker logout ${CI_REGISTRY} &&
    export DEPLOYED_COMMIT_SHA=${CI_COMMIT_SHA} &&
    docker-compose up -d &&
    exit
  environment:
    name: production
  when: manual
  only:
    - master
  tags:
  - deploy
```