TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Roomet Oja 179662IABB

Massiline toodete sisestamine ja uuendamine SAP HANA andmebaasisüsteemis, kasutades MS Excelis olevaid tooteandmeid

Bakalaureusetöö

Juhendajad: Rivo Lemmik
PhD
Jakob Jõgi
Magistrikraad

Tallinn 2024

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: Roomet Oja

03.01.2024

Annotatsioon

Antud töö eesmärk oli luua funktsionaalsus, mis aitaks autori töökohas kasutusel olevasse SAP HANA andmebaasisüsteemi massiliselt importida ja uuendada MS Excelis olevaid tooteandmeid. Lisafunktsionaalsusena peab olema kasutajal võimalik ettevõtte infosüsteemist tooteandmeid eksportida MS Excel faili.

Enne funktsionaalsuse arendamist tutvuti andmete valideermise heade tavadega, kaardistati nõuded ja planeeriti kasutajaliidese ülesehitus.

Töö tulemusena valmis funktsionaalsus koos kasutajaliidesega, mis aitab kasutajal MS Excel failist importida tooteandmed SAP HANA andmebaasisüsteemi, importimise käigus valideeritakse andmed ning protsessi lõppedes antakse kasutajale tagasisidet. Lisaks on kasutajal võimalik tooteandmeid andmebaasisüsteemist eksportida MS Excel faili.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 6 peatükki, 14 joonist, 4 tabelit.

Abstract

Mass Import and Update of Products into SAP HANA Database System Using Product Data from MS Excel

The aim of this work was to develop a solution to mass import and update products into SAP HANA database system, which is used at author's work, using data from MS Excel files. As an add-on it needs to be possible to export data from application to MS Excel file.

Before developing functionality author studied best practices of data validation, mapped functionality requirement and visualized user interface.

As a result of the thesis a functionality with user interface was developed. The solution helps user to import product data from MS Excel file to SAP HANA database, during the import process data is validated and as a result an overview of the processes' results are displayed. It is also possible to export data from application to MS Excel file.

The thesis is in Estonian and contains 35 pages of text, 6 chapters, 14 figures, 4 tables.

Lühendite ja mõistete sõnastik

CRUD	(C-create, R- read, U-update, D-delete) - Andmebaasidega seotud põhioperatsioonid.
ERP	*Enterprise Resource Planning*, majandus- ja juhtimistarkvara.
SQL	*Structured Query Language*, keel andmebaasipäringute kirjutamiseks.
JSON	*JavaScript Object Notation*, JavaScripti objektide notatsioon, inim- ja masinloetav formaat andmete vahendamiseks.
Product owner	Isik arendusprotsessis, kes on vastutav tehtud töö tulemuste osas.
XLSX	Failitüüp, peamiselt Microsoft Excelile.
URI	*Unified Resource Identifier*, ühtne ressursi-identifikaator, ressurssi identifitseeriv märgistring

Sisukord

1 Sissejuhatus 10
2 Kirjanduse ülevaade 12
2.1 Ülevaade ERP süsteemidest 12
2.1.1 SAP HANA 13
2.2 Andmevalidatsioon 15
2.2.1 Eesmärk 15
2.2.2 Andmekvaliteedi tagamise põhimõtted 15
2.2.3 Mitmekihiline veahaldus 16
2.3 MS Excel sisendandmete hoidlana 17
3 Metoodika 19
3.1 Kasutusel olevad tehnoloogiad 19
3.2 Nõuded arendatavale funktsionaalsusele 20
3.2.1 Tooteandmete valideermine 20
3.2.2 Eksport 20
3.2.3 Import 21
3.3 XLSX dokumendi genereerimine 22
3.3.1 Dokumendigeneraatori valimine 22
3.4 XLSX dokumendi sisselugemine 24
3.4.1 XLSX dokumendilugeja valimine 24
3.5 Kasutajaliides 25
3.6 Arenduspõhimõtted 25
4 Teostus 26
4.1 Struktuur 26
4.2 Arendusprotsess 31
4.2.1 Eksport funktsionaalsuse arendamine 31
4.2.2 Import funktionaalsuse arendamine 33
4.2.3 Kasutajaliidese arendamine 38
4.3 Testimine 40
5 Tulemused ja analüüs 41

5.1 Tulemused 41
5.2 Analüüs 42
5.3 Edasised võimalused 43
6 Kokkuvõte 44
Kasutatud kirjandus 45
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 48
Lisa 2 – Näide genereeritud xlsx dokumendist 49

Jooniste loetelu

Joonis 1. Ülevaade SAP HANA DB arhitektuurist [7]. 14
Joonis 2. Andmestruktuuri Entity Relationship diagramm 27
Joonis 3. Backend meetodi kasutamine andmeobjektide väljastamiseks 32
Joonis 4. Eksportimiseks kasutatava meetodi tagastatav objekt..... 32
Joonis 5. Failigeneraatori kasutamine. 33
Joonis 6. FileUploaderi kasutamine File Exploreri avamiseks. 34
Joonis 7. Importimisel andmete lugemine ja struktureerimine..... 34
Joonis 8. Import andmepäringute koostamine 35
Joonis 9. Unikaalsete väärtuste leidmine importimisel 36
Joonis 10. Andmekirjete otsimine andmebaasist unikaalsete väärtuste põhjal. 36
Joonis 11. Import protsessi *backend* funktsionaalsuse tulemina tagastatav andmeobjekt 38
Joonis 12. Dialoogiaken import faili valimiseks 39
Joonis 13. Import protsessi ülevaate dialoogiaken 39
Joonis 14. Dialoogiaken import protsessi tulemuste kuvamiseks 40

Tabelite loetelu

Tabel 1. Failigeneraatorite võrdlus 24
Tabel 2. Ülevaade imporditava toote väljadest 30
Tabel 3. Ülevaade imporditavate tooteseoste väljadest 31
Tabel 4. Eksporditavate andmete koostamise päring 32

1 Sissejuhatus

Kaasaegne ettevõtlus tugineb tehnoloogiale ja andmehaldusele. Üha enam liigutakse digitaliseeritud ja automatiseeritud lahenduste suunas. Taolised lahendused aitavad hallata väga mahukaid andmehulki, tagada andmete valiidsust ning muuta inimressursile ajamahukad tööülesanded oluliselt kiiremaks. Sageli on probleemseks kohaks algandmete sisestamine soovitud süsteemi. Manuaalse sisestamisega võib kaasneda suur ajakulu. Lisaks on manuaalse sisestamise käigus tekkivate vigade oht põhjendamatult suur. Neil põhjustel soovivadki edumeelsed ettevõtted andmesisestust võimalikult suurel määral automatiseerida. Digitaliseeritud andmeid on parem hallata, korduvkasutada, muuta jne.

Autori töökohas on uute klientide lisandumisega üha enam kerkinud esile probleem, kuidas saada tooteandmed ettevõtte süsteemi. Tegemist on peamiselt ehitusvaldkonna toodetega, mida edasises äriprotsessis korduvalt kasutatakse. Klientidel on olemasolevate tooteandmete hoiustamiseks erinevaid lahendusi. Parimal juhul on andmed juba mõnda andmebaasisüsteemi sisestatud. Teisel levinud juhul hoitakse neid MS Excel failides.

Käesoleva töö eesmärgiks on arendada funktsionaalsus tooteandmete importimiseks ja vajadusel uuendamiseks MS Excel failist SAP HANA andmebaasisüsteemi. Andmete importimisel tuleb nende õigsust valideerida, kuvada kasutajale protsessi kulgu ning tegevuse lõppedes tuua esile ülevaade imporditud toodete kogusest ning vigade esinemisel viidata vigaste andmete asukohale. Lisaks tuleb luua funktsionaalsus SAP HANA andmebaasisüsteemist tooteandmete eksportimiseks MS Excel faili.

Soovitud tulemuseni jõudmiseks on autoril plaanis tutuvuda erialase kirjandusega, et leida parim lahendus andmehulga valideerimiseks. Seejärel arendab autor funktsionaalsuse, mis võimaldab infosüsteemi kasutajal valida MS Excel faili ning importida sealsed tooted ettevõtte SAP HANA andmebaasissüsteemi. Lisaks on autoril plaanis luua võimekus ettevõtte süsteemis olevate tooteandmete eksportimiseks, mis

võimaldab kasutajal teostada massilisi muudatusi andmetes ning seejärel uuesti tooted importida.

Töö tulemusena jõuab autori loodud lahendus ettevõtte infosüsteemi *live*- keskkonda ning on kõigile kasutajatele kättesaadav ja kasutatav.

Töö algab sissejuhatusega. Seejärel viiakse läbi ülevaade olemasolevast kirjandusest. Teoreetilises osas kaardistatakse täpsed funktsionaalsuse nõuded ning valitakse autori arvates parim lahendus andmete valideerimiseks. Praktilises osas arendatakse funktsionaalsus koos kasutajaliidesega. Lahenduse valmimisel analüüsitakse ja testitakse funktsionaalsuse tööd ning efektiivsust.

2 Kirjanduse ülevaade

Selles peatükis annab autor ülevaate ERP süsteemidest, tutvustab SAP HANA andmebaasisüsteemi ning toob kirjanduse põhjal välja andmete valideermise häid tavasid.

2.1 Ülevaade ERP süsteemidest

ERP-süsteemide näol on tegemist on majandus- või äritarkvaraga, mis on loodud äriprotsesside lihtsustamiseks ja optimeerimiseks. Algselt keskenduti tootmisprotsessidele, sealhulgas tootmisplaani, varude haldus ja tootmise teostussüsteemidele. Hilisemalt on ERP-süsteemid laienenud, hõlmates teisigi ärivaldkondi, sealhulgas kvaliteedikontrolli, finantseerimist, tootearendust, turundust, müüki, hankimist ja muud. ERP-süsteemi üks põhilisi eesmärke on kriitliste äriprotsesside automatiseerimine, olles samas ka ühtne andmebaas kõigi ettevõtte tegevuste tarbeks. Sel moel on tagatud andmete võimalikult kiire kättesaadavus ning välistatud asünkroonsus. Kõige enam võidavad ERP-süsteemide kasutamisest suured ettevõtted, kus äriprotsessid on väga keerukad, kasutajaid on palju ning enamus protsessidel on sarnane kulg. Mida keerukam on äriprotsess, seda pikemaks ERP-süsteemi juurutamine kujuneb. Keerukamatel juhtudel võib selleks kuluda aastaid [1], [2].

Üldjuhul on ERP-süsteemid modulaarsed, mis tähendab, et kasutajal on ise võimalik vajaduspõhiselt mooduleid valida. Siiski tasub ERP-süsteemi valikul väga põhjalikult hinnata ettevõtte vajadusi ja nõudeid. Kasutatav süsteem peab suutma pakkuda tuge põhilistele äriprotsessidele. Eesmärk on viia võimalikult suur osa tegevustest ühtsesse süsteemi ning vältida olukorda, kus osa toiminguid toetuvad jätkuvalt Excelis paiknevatele andmetele. Valitud moodulid suhtlevad omavahel ning moodustavad ühtse terviku [3].

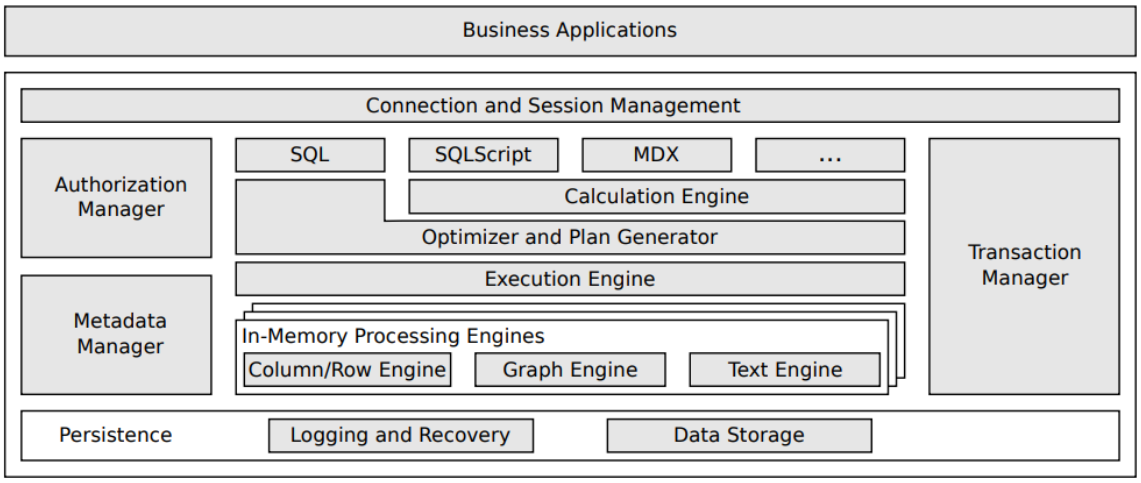
ERP-süsteemi valikul tuleb põhjalikult kaaluda ettevõtte praeguseid ja tulevaseid vajadusi. Tõenäoliselt jääb valitud süsteem kasutusele aastateks ning läbimõtlemata otsus võib endaga kaasa tuua hulganisti probleeme, näiteks halb skaleeritavus, tugiteenuse puudumine. Peamised tegurid, millele valiku käigus mõelda: vajaduspõhisus,

kasutusmugavus, skaleeritavus, kuluefektiivsus, tugiteenuse pakutavus. Ettevõtte algfaasis on sobivat süsteemi kergem leida. Esialgu on võimalik valida põhiprotsesse toetav ERP. Hilisema kasutamise käigus on skaleerimisel võimalik äriprotsesse suunata, et need ühilduksid juba kasutusel oleva süsteemiga. Kui aga suurettevõte soovib haldustarkvara vahetada, või pole mingil põhjusel selle kasutamiseni veel jõudnudki, on olukord kindlasti keerukam. Sageli muutuvad äriprotsessid nii keerukaks ja spetsiifiliseks, et *out of the box* lahendust ei pruugigi olla võimalik leida. Sel juhul tuleks hakata mitut süsteemi omavahel liidestama või lasta vajadustele vastav tarkvara luua. Valmislahenduse kasutuselevõtu kasuks räägivad mitmed tegurid, näiteks kiire juurutamine, arenduskulude puudumine, optimeeritud kasutusprotsess jne [4].

Hästi töötav ERP-süsteem annab kasutajale konkurentsieelise või vähemalt aitab teistega sammu pidada. Praegusel ajal on tarneahelate muutumine, töögruppide restruktureerimine, hinnapoliitika korrigeerimine ja muu taoline täiesti igapäevane praktika. Korrektselt töötav haldussüsteem aitab ettevõttel viia muudatused ellu nii, et sellega ei kaasne seisakuaega ega inimressursi hõivet [4].

2.1.1 SAP HANA

SAP HANA on platvorm, mis koosneb relatsioonilisest andmebaasist, veebiserverist ja teistest andmehaldusmootoritest, mis moodustavad ühtse analüütilise platvormi [5]. SAP HANA andmebaasi üldine eesmärk on pakkuda mälukeskset andmehaldusplatvormi, mis toetab puhtalt SQL-i traditsiooniliste rakenduste jaoks ning lisavõimekust SQLScripti näol spetsialiseeritud SAP rakenduste tarvis. Antud töö kontekstis kasutatakse SAP HANA süsteemi andmebaasina. Süsteemi põhikomponendiks on põhimälul töötavad protsessimootorid. SAP HANA võimaldab andmehulkade rea- või veerupõhist salvestamist, mis on optimeeritud tänapäevaste tarkvarade poolt, et võimaldada paralleelset töötlust [6], [7]. Joonisel 1 on kujutatud SAP HANA üldist andmebaasisüsteemi arhitektuuri.



Joonis 1. Ülevaade SAP HANA DB arhitektuurist [7].

Connection and Session Management loob ja haldab klientide andmebaasisessioone. Juhul kui ühendus on loodud, saab klient andmebaasiga suhtlemiseks kasutada SQL-i, SQLScripti või mõnda muud toetatud keelt. Juhul kui kasutatakse mõnda teist keelt peale SQL-i, toimub keele transleerimine *Calculation Engine* protsessis [6].

Transaction Manager koordineerib andmebaasi transaktsioone ning peab järge käimasolevate ja lõpetatatud transaktsioonide üle [6].

Kliendi päringute töötlemine ja optimeerimine toimub *Optimizer and Plan Generator* kihis. Tulenevalt loodud käivitusplaanist käitab *Execution Engine* mälusiseseid töötlusmootoreid [6].

Authorization Manager kutsutakse välja teiste HANA andmebaasi komponentide poolt, kontrollimaks, kas kasutajal on soovitud protsessi jaoks vajalikud õigused olemas [6].

Metadata Mangeris defineeritakse tabelid, vaated, indeksid ja SQLScripti funktsioonid [6].

Uue andmetabeli defineerimisel määratakse kas kasutatakse rea- või veerpõhist salvestamist. Tulenevalt lähenemisest, et veerupõhised andmed on tugevalt kokkupakitud, mis tagab nende kiire töötluse ja samas hoiab mälusiseste protsesside andmemahtu võimalikult madalana, on mõistlik kasutaja ja rakendusega seonduvaid andmeid hoida veerupõhiselt, eesmärgiga saada võimalikult palju kasu andmete kokkupakitusest ja optimeerimisprotsessist. Metaandmete ja üldiselt harva kasutatavate andmete salvestamiseks kasutatakse enamasti reapõhist salvestamist [6].

Persistance kiht hoolitseb transaktsioonide töökorra, tulemuste logimise ning andmebaasi taastamise eest, juhul kui süsteem näiteks taaskäivitatakse [6].

2.2 Andmevalidatsioon

Andmete valideerimise näol on tegemist protsessiga, mis viiakse läbi andmete kogumise, struktureerimise või analüüsimise käigus, veendumaks, et andmed vastavad nõuetele, ei sisaldaks vigaseid kirjeid ning oleksid võimalikult täpsed. Kvaliteetsed andmed on aluseks edasiste äriprotsessidele edule [8]. Protsessi tulem on iseenesest lihtne, hulk andmeid kas vastab etteantud parameetritele või mitte. Tulenevalt sellest tehakse järeldus, kas andmehulk on kasutatav või tuleks see kõrvale jätta.

2.2.1 Eesmärk

Üldjuhul on andmetele esitud selged nõuded, millised need peavad olema või siis vastupidi, et millised need ei tohi olla. Äritarkvara puhul on vajalik, et kasutaja saaks olla kindel, et kasutusel olevad andmed ka neile reeglitele vastaksid. Kvaliteetsed andmed tagavad, et äriprotsessidel oleksid korrektsed tulemused.

Üks väga oluline valideerimise eesmärke on leida üles kasutaja poolsed vead andmete sisestamisel. Inimesel on andmesisestuse käigus tehtavate vigade esinemise tõenäosus küllaltki suur. Mida suurem on sisestatav andmehulk, seda tõenäolisem on, et kuskil on tehtud mõni viga. Andmete valideermise eesmärk ongi tekkinud vead üles leida ja süsteemi jõudmist takistada. Edasine sõltub juba konkreetsest süsteemist, kas viga õnnestub parandada automaatselt või tuleb kirje kõrvale jätta ja kasutajale sellekohane teave edastada [9].

2.2.2 Andmekvaliteedi tagamise põhimõtted

Peamiselt jagatakse andmete valideerimine kaheks. Üks suundadest on süsteemsetele nõuetele vastavus. Nende nõuete näol on enamasti tegemist infosüsteemist tulenevate reeglitega. Näiteks peavad sisendandmed olema kindlat tüüpi- number, tekst, *boolean.* Lisaks on andmebaasidel ja erinevatel programmeerimiskeeltel omad nõuded numbriliste väärtuste suurusele ja tekst tüüpi väljade pikkusele. Seda tüüpi piirangute näol on tegemist rangete nõuetega ehk neile nõuetele peavad andmed vastama [9].

Teine osa nõuetest on ärilistele nõuetele vastavus. Nende näol on enamasti tegemist äriloogiliste nõuetega andmetele, millele sisendandmed vastama peavad. Äriloogiliste nõuete alla kuuluvad ettevõtte äriprotsessist tulenevate nõuete vajadused [9]. Lihtsustatud näitena peab tootel olema määratud tootekood ja nimi.

Nii äriloogiliste kui ka süsteemsete nõuete näol on üldjuhul tegemist rangete nõuetega ehk seatud nõuded peavad alati olema täies mahus täidetud.

2.2.3 Mitmekihiline veahaldus

Erinevate süsteemide liidestamisel ning seeläbi andmete liigutamisel ühest süsteemist teise on andmete valideermine väga tähtsal kohal. Hea tava järgi on valideerimine ja ühtlasi veahaldus mõistlik jagada erinevatesse kihtidesse. Kihilise struktuuri eesmärk on lihtsustada ja optimeerida valideerimise protsessi. Ühtlasi muudab kihiline lähenemine keeruka valideerimisprotsessi lihtsamini hallatavaks ning aitab kogu protsessi kulgu paremini läbi mõelda.

Kihilise struktuuri puhul alustatakse kõige üldisemast ning liigutakse järk-järgult üha spetsiifilisemate nõuete suunas. Juhul kui konkreetne andmekirje ei vasta ühele kihis olevale nõudele, ei kontrollita kirjet järgnevates kihtides, vaid märgitakse koheselt ebasobivaks ning liigutakse edasi järgnevate kirjete juurde. Olukorras, kus kirje läbib kõik kihid, märgitakse andmekirje sobivaks ning sisestatakse soovitud süsteemi.

Ühe võimaliku lahendusena veahalduse kihtidesse jagamiseks on lähenemine, kus luuakse neli eritasemelist kihti.

Kõige üldisema kihina on defineeritud *Service Level Monitoring* kiht. Selle kihi peamiseks ülesandeks on süsteemi töökindluse ja võimekuse kontrollimine. Kontrollitakse, et vajalikud teenused on kättesaavad. Töötava süsteemi puhul ei tohiks selles kihis vigu esineda, siiski tekkinud vigade korral tuleks neid juhtumipõhiselt analüüsida.

Sellele järgneb *Process Level Error Handling*. Püütakse teadaolevaid, kuid harva esinevaid vigu. Eesmärgiks on tööprotsesside üksteisest eraldamine. Sisuliselt on tegemist kapseldamisega. Peamiselt kasutatakse *try/catch* veahaldust, mis püüab kinni vea saanud tööprotsessi ning liigub edasi järgmise protsessi suunas. Selles kihis ei pöörata tähelepanu äriloogilistele vajadustele, ehk sageli tekib olukord, kus kinni püütud protsessi

tulem oleks vajalik järgneva protsessi läbiviimiseks ning seetõttu tekibki *Transaction Level Exception* taseme viga järgnevas protsessis. Tavalisteks tekkepõhjusteks on vigased veebipäringud, ebakorrektsed sisendandmed vms.

Transaction Level Exception Handling kihis lähenetakse tulenevalt äriloogikast. Kõik teadaolevad vigade tekkekohad on defineeritud ning vea tekkimisel üritatakse läheneda alternatiivse valiku läbi. Veakoodid on samuti määratletud ning vea tekkimisel, mille tulemusena ei ole võimalik protsessiga jätkata, tagastatakse kindlaksmääratud veakood ning tööprotsess peatub. Peamiselt on tegemist andmekvaliteedi ja äriloogikast tulenevate andmevalidatsiooni vigadega.

User Level Intervention kihi näol on tegemist kasutaja tasemel veahalduse kasutajaliidesega, mida kasutatakse peamiselt vigade analüüsimiseks, logide vaatamiseks, statistika kogumiseks jne. See annab ülevaate süsteemis tekkinud vigadest ja nende põhjustest ning on sisendinfoks edasistele süsteemi arendustele ja vigaste funktsioonide parandamisele [10].

Neid põhimõtteid järgides on veahaldus võimalikult efektiivne ning aitab ära hoida tõrkeid süsteemi töös.

2.3 MS Excel sisendandmete hoidlana

Tänapäeval kasutavad jätkuvalt paljud ettevõtted äriprotsessides käigus Excelit, kohati on see põhjendatud, kuid peamise andmebaasina Exceli kasutamist ei peeta heaks tavaks. Õnneks on paljud ettevõtted siiski innovatiivsed ning soovivad liikuda ühtsete infosüsteemide kasutamise suunas. Massiivsete andmehulkade haldamine Excelis eeldab väga head ülevaadet kogu struktuurist. Seoste haldamine muutub keerukaks ning mitmete kasutajate puhul ei suudeta andmekvaliteeti ja andmete õigsust tagada. Samas on MS Exceli näol tegemist vägagi võimeka tööriistaga ning kohati on tegemist asendamatu abilisena töötajale [11].

MS Exceli head omadused andmete hoiustamisel:

- Väiksemate andmehulkade puhul mugav kasutada ja annab hea ülevaate andmetest.

- Lihtne kasutada. Enamik inimesi on Exceliga kokku puutunud ja oskavad mõningal määral süsteemi kasutada.

- Palju erinevaid funktsioone. Excel omab väga laia funktsionaalsuste valikut, andmehulki on lihtne analüüsida, muuta ja korrastada.

MS Exceli võimalikud probleemsed kohad andmete hoiustamisel, millele tasub tähelepanu pöörata:

- Probleemid seoste haldamisega. Relatsiooniliste andmebaaside puhul on üldjuhul seoste haldamine mingil moel lahendatud, kuid Excelis on üksteisest sõltuvate andmehulkade defineerimine ja haldamine palju keerukam, kuna puudub otsene lahendus struktuursete andmete loomiseks.

- Piiratud mahutavus. Suurte andmehulkade puhul võib Excel muutuda aeglaseks.

- Andmete turvalisusega seonduvad ohud. Excelis puudub hea lahendus andmehulkadele ligipääsu seadmiseks.

- Andmekvaliteedi potentsiaalne risk. Excelis andmehulkadele nõuete seadmine on keerukam kui andmebaasisüsteemides. Sisendandmete nõuetele vastavuse kontrollimine ei ole efektiivselt lahendatav.

Kokkuvõtvalt on Exceli näol tegemist väga kasuliku ja võimeka rakendusega, kuid andmete järjepideval hoiustamisel on vead kerged tekkima ja andmekvaliteedi tagamine muutub keerukaks.

Kuna MS Excel on tänapäeval väga laialdaselt kasutusel, on selle võimaluste valik äärmiselt lai. Juhul kui soovitakse integreerida Excelis paiknevaid andmeid mõnda teise andmebaasisüsteemi, on teatavad valikud olemas. Näiteks importimise ja eksportimise kontekstis on võimalikud abilised *Power Query* ja *XML data source.*

Power Query on andmete transformeerimise ning ettevalmistamise mootor. *Power Query* sisaldab graafilist kasutajaliidest andmete hankimiseks erinevatest allikatest ning redaktorit transformatsioonide rakendamiseks. Konkreetse töö kontekstis võimaldaks *Power Query* luua ühenduse MS Excel faili ja SAP HANA andmebaasi vahel ning seeläbi võimaldada reaalajas andmebaasis paiknevate andmete muutmist MS Excel failis.

Tulenevalt töös kasutatava infosüsteemi autoriseerimise eripäradest ja veahalduse implementeerimise soovidest, jäeti konkreetne võimekus sel korral kõrvale [12].

XML data source on tehnoloogia, mis on loodud struktureeritud andmete haldamiseks tekstifailis. XML järgib standardjuhiseid ning võimaldab töötlemist mitmesuguste andmebaaside ja rakenduste abil. XML võimaldab luua kohandatud vorme, andmestruktuure ja skeeme. Kokkuvõttes hõlbustab XML oluliselt andmete määratlust, edastamist, valideerimist ja tõlgendamist andmebaaside, rakenduste ja organisatsioonide vahel [13]. Antud töö konteksis võimaldaks eeldefineerida seosed andmebaasitabeli väljade ja MS Excel faili lahtrite vahel, mis hõlbustaks andmete paigutamist xlsx failis ning mõneti ka sisselugemisel *mappingute* loomisel. Samas eeldab XML vormingus andmete edastust ja vormide loomist.

3 Metoodika

Selles peatükis kirjeldab autor funktsionaalsuse nõudeid, analüüsib dokumendi genereerimise ja sisselugemise teenuseid, toob välja kasutajaliidese vajalikud võimalused ning tutvustab arendamise käigus järgitavaid põhimõtteid.

3.1 Kasutusel olevad tehnoloogiad

Autor lähtus arendatava funktsionaalsuse loomisel peamiselt juba ettevõtte infosüsteemis kasutusel olevatest tehnoloogiatest ja seeläbi seatud piirangutest.

Andmebaasisüteemina kasutati SAP HANA andmebaasi haldussüsteem. Andmebaasiartefaktide hoidmiseks kasutati HDI konteinereid ning *backend* arenduskeskkonnana toetuti SAP Web IDE keskkonnale. Arenduskeskkonnas oli kasutusel Node.js käitlussüsteem, mis kasutab Javascript programmeerimiskeelt.

Rakenduse visuaalse poole haldamiseks kasutati kasutajaliideste raamistikku SAPUI5. Kasutajaliidese loogika juhtimiseks kasutati Javascript kontrollereid ning visuaalse poole kuvamiseks XML vaateid. Kasutajaliidese disainimisel lähtuti SAP Fiori Guideline juhtistest ning parimatest praktikatest.

Koodilahenduse versioonide haldamiseks kasutati Bitbucket tarkvara. *Frontend* arenduskeskkonnana kasutati Visual Studio Code tarkvara.

3.2 Nõuded arendatavale funktsionaalsusele

Autor kaardistas koostöös ettevõtte *product owneriga* vajalikud nõuded arendatavale funktsionaalsusele. Nõuete kaardistamisel lähtuti klientide soovidest ja olemasoleva andmebaasistruktuuri piirangutest ja võimalustest. Alustati üldise funktsionaalsuse nõuete paikapanemisest ning liiguti järk-järgult spetsiifilisemaks. Aeg-ajalt suheldi kliendiga ja valideeriti planeeritavat lahendust.

Põhifunktsionaalsusena peab kasutajal olema võimalik MS Excelis olevaid tooteandmeid importida ettevõtte SAP HANA andmebaasisüsteemi ja ka vastupidi ehk eksportida SAP HANA andmebaasisüsteemist MS Excel faili. Kasutajal peab olema ülevaade protsessi kulgemisest ning lõpptulemina tehtud edusammudest.

3.2.1 Tooteandmete valideermine

Arendatava funktsionaalsuse põhirõhk langeb andmete valideermisele. Eksportimise protessi käigus peavad andmed saama väljastatud samal kujul nagu need olid ettevõtte andmebaasis. Eksportimisel eraldi andmete nõuetele vastavust kontrollida ei tule, sest andmebaasi sisestamisel on kõik kirjed nõutud kontrollid läbinud.

Toodete importimisel tuleb kontrollida kõiki äriloogikast ja andmebaasisüsteemist tulenevaid nõudeid. Esmalt tuleb kontrollida failitüübi vastavust, et välistada kasutajate valede failide süsteemi sisestamise võimalus. Sobiva failitüübi korral liigutakse edasi andmehulga kontrollimise juurde. Tooteandmetele peavad importimisel rakenduma kontrollid, mis tagavad samaväärse andmekvaliteedi süsteemisisese toote loomise või uuendamise protsesside korral.

3.2.2 Eksport

Kasutajapõhised funktsionaalsed nõuded tooteandmete eksportimisele ettevõtte infosüsteemist MS Excel faili:

- Kasutajana soovin eksportida kõik kättesaadavad tooted infosüsteemist MS Excel faili.

- Kasutajana soovin, et kõigile toodetele oleks lisatud konkreetse toote parameetrid ja klassifikaatorid.

- Kasutajana soovin näha toodetega seotud toodete loetelu.

Mainitud nõuded tagavad ekspordi kasutatavuse süsteemi kasutajatele. Tulenevalt ekspordi protsessist ja klientide nõuetest piisab konkreetse arenduse käigus väljatoodud nõuete täitmisest.

3.2.3 Import

Kasutajapõhised funktsionaalsed nõuded tooteandmete importimisele MS Excel failist ettevõtte infosüsteemi:

- Kasutajana soovin sisendandmeid sisaldava andmefaili valimist.

- Kasutajana soovin luua uusi tooteid.

- Kasutajana soovin uuendada olemasolevaid tooteid.

- Kasutajana soovin siduda toote teiste toodetega.

- Kasutajana soovin näha reaalajast ülevaadet protsessi kulgemisest ja tulemustest.

- Kasutajana soovin võimalust peatada importimise protsess igal hetkel.

- Kasutajana soovin protsessi lõppedes näha ülevaadet tulemustest- kui palju tooteid sisestati, kui palju tooteid ei õnnestunud sisestada.

- Kasutajana soovin tagasisidet viidetega vigaste andmete asukohale.

- Kasutajana soovin võimalust valida, kas importida toodete omavahelised seosesed või mitte.

Mainitud nõuete täitmine on vajalik kasutajatele tervikliku teenuse pakkumiseks. Nõuded valideeriti potentsiaalsete kasutajatega ja jõuti järeldusele, et esialgse arendusprotsessi käigus nende täitmisest piisab.

3.2.3.1 Toote loomine

Uue toote süsteemi sisestamise eelduseks on unikaalne tootekood. Tulenevalt relatsioonilisest andmebaasistruktuurist on ainus uus loodav andmeobjekti kirje toode ise. Kõik tootega seonduvad objektid nagu klassifikaatorid, parameetrid, ühikud jne, peavad olema eelnevalt andmebaasissüsteemis defineeritud ning nende puhul luuakse seos toote ja andmeobjekti vahel, kuhu sisestatakse ka soovitud väärtus.

3.2.3.2 Toote uuendamine

Toote uuendamise eelduseks on identse tootekoodiga toote olemasolu andmebaasis. Sarnaselt toote loomisele ei looda uuendamise käigus andmebaasis uusi andmeobjekte, vaid uuendatakse vastavaid välju või vahetabelites paikenevate seosete väärtuseid.

3.3 XLSX dokumendi genereerimine

Andmete põhjal kindlat tüüpi dokumentide loomiseks leidub palju erinevaid teenuseid, mis suudavad sisendandmete ja vajalike parameetrite etteandmisel automaatselt kujundada ja genereerida soovitud failitüübiga faili. Antud töö käigus on autoril soov kasutada funktsionaalsust xlsx failitüübiga faili loomiseks. Taoline lahendus tagab kliendile võimaluse vajutada nupule, mille tagajärjel käivitatakse meetod, mis pärib andmebaasist vajalikud andmed, struktureerib andmed vastavusse generaatori nõuetega ning edastab andmehulga generaatorile. Dokumendigeneraator tagastab soovitud faili kasutaja soovitud andmetega.

3.3.1 Dokumendigeneraatori valimine

Esialgu kaardistas autor koostöös *product owneriga* vajalikud nõuded generaatorile. Kuna ettevõtte eesmärk on luua lisakulu mittenõudev lahendus, on vajalik, et kasutatav generaator oleks tasuta kasutatav. Dokumendigeneraatori nõuded tulenevad otsesest vajadusest protsessi soovitud kujul läbiviimiseks.

Nõuded generaatorile:

- Generaator peab suutma struktureerida etteantud andmehulga maatriksi kujule, kus veeru päistes on nimetus ja ridadel kuvatakse nimetusele vastavat väärtust.
- Generaator peab suutma genereerida mitme vahelehega dokumendi.

- Generaator peab suutma defineerida andmefaili lahtri tüüpe. Näiteks tekst, number, *boolean* tüüpi lahtreid.

Generaatori valikul tuli lähtuda olemasolevate tehnoloogiate piirangutest ning arvestada vajadusega, et integreerimine kasutusel olevate lahendustega oleks võimalikult lihtne ja jätkusuutlik. Autoril õnnestus tehnoloogiatega tutvumisel leida kaks potentsiaalset lahendust. SheetJS [14] poolt pakutav lahendus ja SAPUI5 enda poolt pakutav Spreadsheet [15] lahendus.

Täpsemal SheetJS ja Spreadsheet lahenduste analüüsimisel selgus, et mõlemal tehnoloogial on positiivsed ja negatiivsed küljed. Spreadsheet implementeerimine on lihtsam ning enamus nõuetele funktsionaalsus ka vastab, kuid mitmele vahelehele andmete paigutamise võimekus puudub. Seega osutus Spreadsheet lahendus ebasobivaks. SheetJS lahendus täidab kõiki soovitud nõudeid. Kahjuks puudub tasuta versioonis veerupäiste esiletoomise ja lahtrivärvi valiku võimalus, kuid nende näol ei olnud tegemist rangete nõuetega ning hilisema vajaduse käigus on ka need funktsionaalsused SheetJS PRO tasulises versioonis saadaval. Tabelis 1 on välja toodud SheetJS ja Spreadsheet lahenduse nõuetele vastavuse võrdlus.

	SheetJS	Spreadsheet
Maatriksi kujul andmete kuvamine	Jah	Jah
Mitme vahelehega dokumendi loomine	Jah, piiramatu arvu vahelehtede loomise võimekus	Ei, võimalus luua üks lisa vaheleht metaandmetele, kuid mitte rohkem
Võimekus defineerida genereeritava lahtri tüüpi	Jah	Jah

Võimekus muuta lahtrite taustavärvi	Jah, kuid ainult tasulises PRO versioonis.	Ei
Litsents	Apache 2.0 License	Kommertslitsents

Tabel 1. Failigeneraatorite võrdlus

3.4 XLSX dokumendi sisselugemine

Olemasolevate tooteandmete sisselugemine xlsx tüüpi failist on kasutajate poolt väga soovitud funktsionaalsus. Sarnaselt dokumendi genereerimisele on andmete lugemiseks samuti mõistlik kasutada kolmanda osapoole loodud funktsionaalsust. Funktsionaalsusele antakse ette xlsx tüüpi fail ning vastusena saadakse failis olevad andmed ridade põhiselt.

3.4.1 XLSX dokumendilugeja valimine

Sarnaselt dokumendigeneraatori valmisele alustati lugeja valikul nõuete kaardistamisest. Dokumendigeneraatori nõuete seadmisel lähtuti peamiselt loodava funktsionaalsuse nõuetest. Generaator peab suutma täita kõiki loodavas funktsionaalsuses vaja olevaid aspekte.

Nõuded dokumendilugejale:

- Dokumendilugeja peab suutma lugeda xlsx failis olevaid andmeid reapõhiselt ning tagastama andmeobjektid, kus veerupäistele vastavad real paikenevad andmed.
- Dokumendilugeja peab suutma lugeda andmed mitmelt vahelehelt.
- Dokumendilugeja peab suutma eristada xlsx failis oleva andmekirje lahtri tüüpi.

Pakutavate lahenduste otsingul tutvus autor esialgu juba tuttavate lahendustega, milleks olid SheetJS ja SpreadSheet. SpreadSheet lahendusel puudus soovitud võimekus, kuid SheetJS täitis vajalikud kriteeriumid. Seega valituks ostus SheetJS. SheetJS lahendus vastas kõigile seatud nõuetele ning kuna dokumentide genereerimiseks planeeris autor

kasutusele võtta sama pakkuja tehnoloogiat. Näis autorile mõistlik kasutada dokumentide sisselugemiseks SheetJS lahendust.

3.5 Kasutajaliides

Kasutajaliidese disainimise käigus tuleb arvestada kasutajamugavustega ning kasutaja teavitamisega protsessi staatusest. Tooteandmete importimise ja eksportimise näol on tegemist väga mahukate ja ressursinõudlike protsessidega, mis võivad suurte andmemahtude käigus ajakulukaks muutuda, kuid näiteks Jakob Nielseni sõnul on kasutaja tähelepanu püsivuse piiriks umbes 10 sekundit [16]. Lähtudes sellest põhimõttest tuleb kasutajaliides arendada vastavalt, et kasutajal oleks reaalajas võimalik näha protsessi kulgu.

Eeldusel, et eksportimise protsess on vähem ajamahukas, võib selle maksimaalseks ajakuluks võtta 10 sekundit. Importimise protsess on aga tulenevalt keerukast arhitektuurist, suurest seoste arvust ja andmebaasi CREATE, UPDATE päringute efektiivsusest ajamahukam ning sel põhjusel ei saa eeldada, et ajakulu üle piiri ei läheks. Seega tuleb lähtuda põhimõttest, et kasutajat tuleb teavitada eeldatavalt pikast ajakulust ning kuvada kasutajale indikaatorit protsessi etappidest ja eeldatavast ajakulust.

Importimise protsessi lõppedes tuleb kasutajale kuvada sisestuste ja tekkinud vigade kohta infot: vigade kogused ja tekkepõhjus. Eesmärgiks on, et kasutajal oleks selge ülevaade sisestatud toodetest ja vigastest andmetest.

3.6 Arenduspõhimõtted

Arenduse käigus tugines autor tarkvaraarenduse headele tavadele, et valmiv lahendus oleks teistele arendajatele võimalikult loetav, töötaks korrektselt ning oleks skaleeritav, samas jõudes töötava lahenduseni võimalikult lühikese aja jooksul. Järgnevalt tutvustab autor mõningaid arendamise käigus järgitud praktikaid, metoodikaid ja häid tavasid.

Extereme programming on üks olulisemaid tarkvaraarenduse raamistikke agiilsete mudelite seas. See keskendub tarkvara kvaliteedi parandamisele ja kiirele reageerimisele klientide nõuetele. Soovitab võtta seni hästi toiminud parimad tavad ning rakendada neid äärmuslikul tasemel [17]. Näiteks kasutas autor paarisprogrammeerimist mõne *senior* arendajaga ning pidevalt liigutati muudatusi test-keskkonda, vigade leidmiseks ja edasiste

arenduste planeerimiseks. Lisaks sellele toimus aegajalt *code review*, mille käigus tutvus ja hindas autori kirjutatud koodi mõni teine arendaja.

Feature-Driven Development näol on tegemist lähenemisega, mis jagab arendatava lahenduse väiksemateks loogiliselt jaotatud tükkideks [17]. Arendamine toimus kahe nädala pikkustes *sprintides*, mille jooksul oli autori tööks töötsükli algul planeeritud funktsionaalsuste arendamine. Sel moel oli tagatud pidev progress ja valmivat lahendust oli võimalik osaliselt testida.

Manage Usage of Third-Party Code on mõtteviis, et olemasolevat lahendust ei ole sageli mõistlik ise uuesti arendama hakata, vaid efektiivsem on kasutada kellegi teise loodud funktsionaalsust, seejuures tuleb aga veenduda lahenduse kvaliteedis. [18] Näiteks kasutas autor dokumentide genereerimisel ja sisselugemisel valmis kolmanda osapoole lahendust, säästes seeläbi palju töötunde.

Keep It Simple, Stupid põhimõtte järgi tuleb arendatavad funktsionaalsuse osad hoida võimalikult lihtsana. Lihtsad lahendused on paremini mõistetavad, muudetavad ja vajadusel parandatavad [17]. Autor üritas funktsionaalsust jagada võimalikult lihtsateks ja loogilisteks osadeks, mis võimaldas loodud meetodeid korduvkasutada ning seeläbi hoida ära liigset koodi duplikatsiooni.

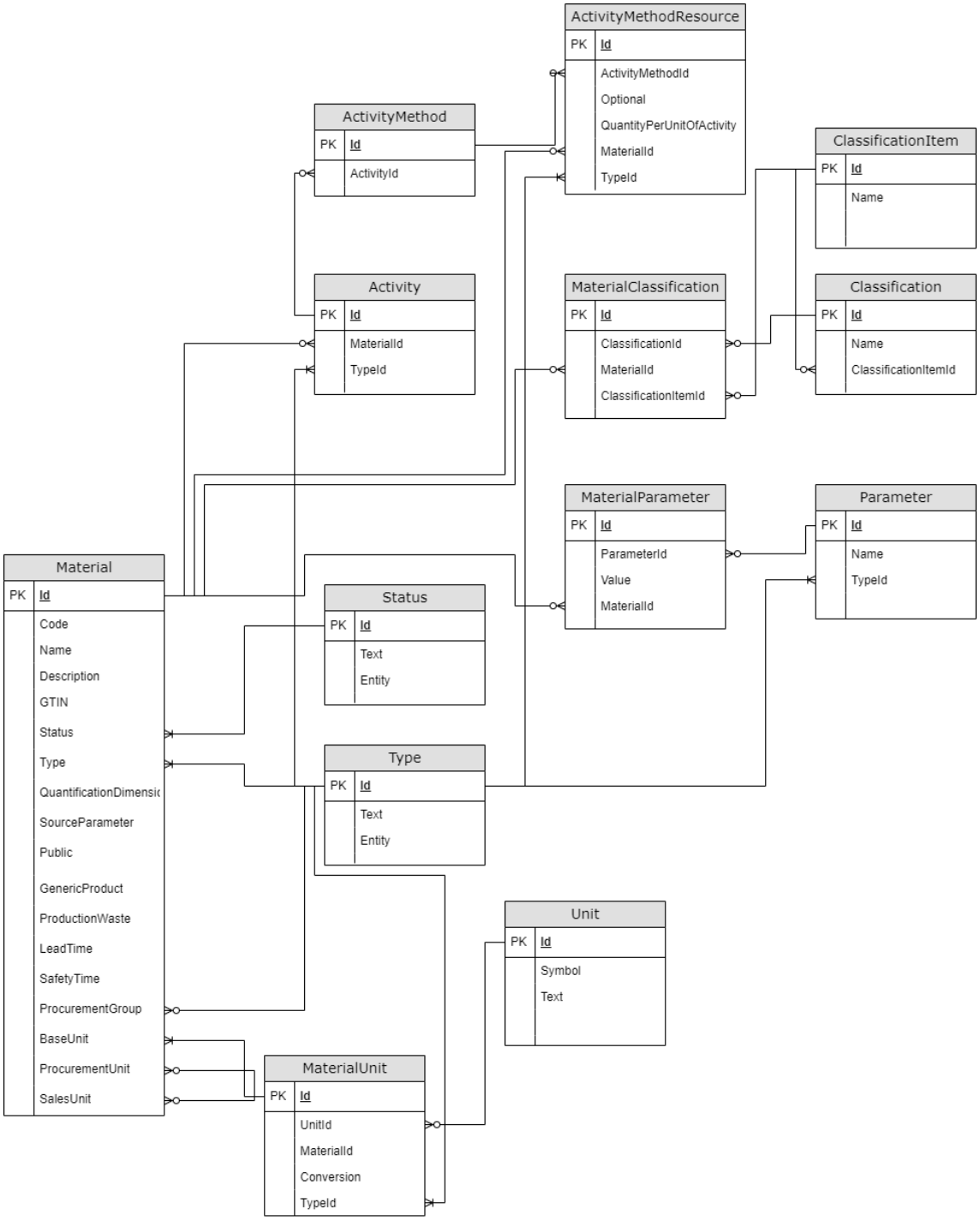
4 Teostus

Selles peatükis tutvustatakse funktsionaalsuse valmimise tööprotsessi, kirjeldatakse koodilahendusi, nõuete täitmist ning lahenduse testimist.

4.1 Struktuur

Andmebaasi andmestruktuuri kirjeldamiseks lõi autor *Entity Relationship* tüüpi diagrammi. Diagrammil on välja toodud töö käigus muudetavate andmetabelite seosed ja nende antud töö konkektstis vajalikud väljad. Ärilistel põhjustel ei ole välja toodud sügavusse minevaid seosetabeleid ja ülejäänud andmevälju tabelites. Samuti on välja

jäetud autoriseerimisega seonduvad tabelid ja andmeväljad. Andmestruktuur on nähtav joonisel 2.



Joonis 2. Andmestruktuuri Entity Relationship diagramm

Lisaks lõi autor tabeli muudetavate andmeväljade kohta ning kirjeldas piiranguid väärtuste sisestamisel. Tabelis 2 on välja toodud xlsx faili esimesel vahelehel sisestavad tooteandmete tulbad ja väärtuste kitsendused. Piirangud ja eeldused väärtustele tulenevad

infosüsteemi andmebaasipiirangutest väärtustele. Import protsessi käigus peavad sisestatavad väärtused vastama samadele nõuetele nagu süsteemis endas sisestamisel.

Nimetus	**Kirjeldus**	**Piirangud ja eeldused väärtustele**
Code	Tootekood	Unikaalne, tühja väärtuse korral tootele määratud seeria järgnev kood.
Name	Toote nimetus	*Not null*
Description	Toote kirjeldus	-
GTIN	*Global Trade Item Number*	-
Status	Toote staatus- aktiivne või mitteaktiivne	Aktiivne või mitteaktiivne
Type	Tootetüüp	Süsteemis defineeritud tootetüüp
Quantification dimension	Toote mõõdistamiseks kasutatav dimensioon	Süsteemis olemasolev dimensioon toodete mõõdistamiseks
Source parameter	Toote mõõdistamiseks kasutatava dimensiooni parameeter, näiteks kõrgus, laius jne.	Süsteemis olemasolev dimensiooni defineeriv parameeter
Public	Toote nähtavus süsteemis. Avalikult kättesaadav või privaatne	Tõene või väär

Generic product code	Baseeruva referentstoote kood	Olemasoleva või import protsessi käigus loodava toote kood.
Production waste	Arvestatav kulu protsentides	Täisarvuline väärtus või *null*
Lead time	Aeg päevades, millega peab arvestama enne toote kasutamist	Täisarvuline väärtus või *null*
Safety time	Säilivusaeg	Täisarvuline väärtus või *null*
Procurement group	Sisseostu koguse arvestamine. Lao põhine või projekti vajaduste põhine arvestus.	Süsteemis eeldefineeritud väärtus kas lao põhiseks või projekti põhiseks vajaduse arvestuseks.
Base unit	Toote koguse ühik	SI-süsteemi kuuluva ühiku tähis
Procurement unit	Ühik toote ostmisel	SI-süsteemi kuuluva ühiku tähis
Procurement unit multiplier	Ostetava koguse kordaja, näiteks juhul kui toodet ostetakse kinda mahuga pakkides	Täisarvuline väärtus, *not null*
Sales unit	Ühik toote müümisel	SI-süsteemi kuuluva ühiku tähis
Sales unit multiplier	Müüdava koguse kordaja, juhul kui toodet müüakse kindlas koguses pakituna	Täisarvuline väärtus, *not null*

Classification_*	Toote klassifikaator	Klassifikaatori definitsioon peab olema süsteemis eksisteeriv, väärtustena valikus võimalikud klassifikaatorid.
Parameter_*	Toote parameeter	Parameetri definitsioon peab olema süsteemis eksisteeriv, väärtuse määramisel tuleb veenduda parameetri tüübi sobivuses.

Tabel 2. Ülevaade imporditava toote väljadest

Tabelis 3 on kirjeldatud tooteseoste võimalikke välju. Tootseoseid defineeritakse xlsx faili teisel vahelehel. Seotud toodete näol on tegemist toodete omavahelise seotuse määramisega, mis tähendab, et olukorras, kus mõne kindla toote näiteks paigaldamise eelduseks on teise toote olemasolu, defineeritakse eelnevalt toodete seos koos vajalike koguste ja ühikutega.

Nimetus	**Kirjeldus**	**Piirangud ja eeldused väärtustele**
Parent Code	Põhitoote unikaalne kood	Vastava koodiga toode peab olema süsteemis olemas
Code	Seotava toote kood	Vastava koodiga toode peab olema süsteemis olemas
Quantity	Seotava toote kogus ühe põhitoote kohta	Arvuline tootekogus
Unit	Seose koguse ühik	SI-süsteemi kuuluva ühiku tähis

Optional	Toote tähtsuse kriitilisus. Kas valikuline või mitte.	Tõene või väär

Tabel 3. Ülevaade imporditavate tooteseoste väljadest

4.2 Arendusprotsess

Peamine arendusprotsess jagati kahte tööetappi. Esimese etapi käigus oli eesmärk võtta kasutusele dokumendigeneraator ning luua võimalus kasutajatel tooteandmeid xlsx faili eksportida. Teise etapi eesmärk oli luua funktsionaalsus toodete xlsx failist SAP HANA andmebaasisüsteemi importimise funktsionaalsus.

4.2.1 Eksport funktsionaalsuse arendamine

Dokumentide loomiseks võeti kasutusele SheetJS [14] tarkvara. SheetJS tasuta versiooni lähtekood on vabalt kättesaadav. Tulenevalt ettevõtte soovist, et kolmanda osapoole kood töötaks muutumatult ja aja jooksul tehtavad muudatused ei mõjutaks olulisel määral kasutatava rakenduse tööd, kopeeriti SheetJS lahenduse lähtekoodi vajalik osa ettevõtte *backend* süsteemi. Kopeerimise hetkel võeti kasutusele versioon 0.20.0. Lisaks sellele võimaldab lähtekoodi kopeerimine vajadusel ka SheetJS lahenduse laiendamist ning piisab vaid kooditäienduste tegemisest.

Generaatori kasutamiseks tuleb rakendusele ette anda xlsx faili sisestatavad JSON kujul andmeobjektid, millest iga objekt sisaldab ühe toote andmeid. Andmeobjektide küsimiseks andmebaasist ja soovitud kujule struktureerimiseks loodi *backend* meetod, mis tagastab generaatorile edastatavad andmeobjektid. Meetod käivitatakse kasutajaliideses eksport nupu vajutamisel. Tabelis 4 on ülevaade tehtavast päringust.

URI	Meetod	Tagastus	Kirjeldus
/api/team/v1/material:excelExport	POST	JSON andmeobjektid toodetest, klassifikaatoritest,	Genereerib andmeobjekti toodetest ja nendega seonduvast, mis

		parameetritest ja seotud toodetest.	edastatakse xlsx faili generaatorile.

Tabel 4. Eksporditavate andmete koostamise päring

Joonisel 3 on näha meetodi kasutamist, sisendina antakse kaasa toote süsteemne tüübikood.

```
return api.post('/api/team/v1/material:excelExport', oBody)
    .then(this.handleExportToExcel.bind(this))
    .catch(err => {
     displayError.displayMessage({
      message: err.message || err.error,
      type: err.type || null
     });
    });
```

Joonis 3. Backend meetodi kasutamine andmeobjektide väljastamiseks

Meetod kogub kokku tooted, mille nägemiseks on kasutajal õigus. Lisaks loetelule toodetest tagastab meetod ka toodetega seotud klassifikaatorid ja parameetrid koos väärtustega ning seotud toodete loetelu. Vigade ilmnemisel antakse kasutajale sellest teada veateatena. Meetodi tagastatav andmeobjekt on nähtav joonisel 4.

```
return {
        aMaterials: aMaterialToExport,
        aMaterialClassifications: aMaterialClassifications,
        aProductParameters: aProductParameters,
        aLinkedProducts: aLinkedProducts
    };
```

Joonis 4. Eksportimiseks kasutatava meetodi tagastatav objekt.

Järgnevalt edastatakse andmeobjektid failigeneraatorile. Samuti on vaja generaatorile ette anda tulpade pealkirjad. Tooteandmete tulba nimena sisestatakse andmeobjektide *key*

väärtused, klassifikaatorite ja parameetrite eristamiseks lisatakse nende tulpade nimedele ette vastavalt Classification_ või Parameter_.

Joonisel 5 on näidatud andmeobjektide edastamist failigeneraatorile ning tooteandmete ja seotud toodete vahelehtede lisamist. Joonise viimasel real on välja toodud ka faili genereerimiseks kasutatav käsk. Peale seda tagastatakse kasutajale xlsx failitüübiga dokument, mille nimeks on Products.xlsx. Eksporditud faili näide on leitav Lisas 2.

```
/* generate worksheet and workbook */
    const worksheetProducts = XLSX.utils.json_to_sheet(aProductRows);
    const worksheetLinkedProducts = XLSX.utils.json_to_sheet(aLinkedProductRows);
    const workbook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(workbook, worksheetProducts,
this.getI18nText('ExportToExcelProductsSheetName'));
    XLSX.utils.book_append_sheet(workbook, worksheetLinkedProducts,
this.getI18nText('ExportToExcelLinkedProductsSheetName'));

    /* fix headers */
    XLSX.utils.sheet_add_aoa(worksheetProducts, [aAllProductColumnNames], { origin: 'A1' });

    /* generate file */
    XLSX.writeFile(workbook, 'Products.xlsx', { compression: true });
```

Joonis 5. Failigeneraatori kasutamine.

4.2.2 Import funktionaalsuse arendamine

Esmalt alustas autor dialoogiakna arendamisega, et võimaldada xlsx faili valikut ning et testida SheetJS andmetelugemise lahenduse töötavust. Kasutajale faili valiku võimaldamiseks kasutas autor SAPUI5 komponenti FileUploader [19], mis võimaldab nupu vajutusel avada Windowsi failihalduri (ingl. k File Expolorer). FileUploaderis on võimalik defineerida, mis tüüpi faile failihalduris kuvatakse. Sel moel on võimalik piirata failide valikut vaid xlsx piires, mis ühtlasi toetab mitmekihilise veahalduse printsiipi, et sisestada saab vaid soovitud tüüpi faile. Joonisel 6 on näidatud FileUploaderi kasutamist.

```
<u:FileUploader id="idFileUploader" name="myFileUpload" change="handleImportComplete"
tooltip="{i18n>ImportDialogFileLable}" sendXHR="true" sameFilenameAllowed="true"
useMultipart="false" fileType="XLSX,xlsx"/>
```

Joonis 6. FileUploaderi kasutamine File Exploreri avamiseks.

Järgnevas faasis loetakse kastutaja failist andmed ning struktureeritakse need soovitud kujule, et edastada valideerimisele. Joonisel 7 on näidatud SheetJS XLSX funktsionaalsuse kasutamist andmehulga failist sisselugemiseks ning struktureerimiseks.

```
reader.onload = e => {
        const data = e.target.result;
        const workbook = XLSX.read(data, {
          type: 'binary',
          cellDates: true
        });
        workbook.SheetNames.forEach(function (sheetName, index) {
          if (index < 1) {
            oData.excelProductsData =
XLSX.utils.sheet_to_row_object_array(workbook.Sheets[sheetName]);
          } else {
            oData.excelLinkedProductData =
XLSX.utils.sheet_to_row_object_array(workbook.Sheets[sheetName]);
          }
        });
        oModelData.excelLinkedProductData = oData.excelLinkedProductData || [];
        oModelData.excelData = oData.excelData || [];
        oModelData.importResult = [];
        this.findImportResults();

        oModel.refresh(true);
      };
```

Joonis 7. Importimisel andmete lugemine ja struktureerimine.

Meetodite tulemusena tagastatakse andmeobjektide hulgad, milles on defineeritud tooted ja tooteseosed struktureeritult edasiseks valideerimiseks ja töötlemiseks.

Tulenevalt nõudest kuvada kasutajale pidevat tagasisidet protsessi tulemustest, jagati andmeobjektid hulga päringute vahel, et saada soovitud infot tulemuste reaalajast kuvamiseks. Joonisel 8 on näha andmehulga tükeldamist ja päringu koostamist.

```
if (oImportDialogModelData.RowsToImport.length) {
      for (let i = 0; i < oImportDialogModelData.RowsToImport.length; i += 
iAmountOfRowsPerRequest) {
        if (oImportDialogModelData.bRequestRunning) {
          const aPortionOfRows = oImportDialogModelData.RowsToImport.slice(i, i + 
iAmountOfRowsPerRequest);
          const oBody = {
            EntityType: 'Product',
            EntityId: 5801,
            aRows: aPortionOfRows
          };

          // Send the API request for each portion and wait for the response before 
proceeding to the next portion
          await api.post('/api/team/v1/material:excelImport', oBody)
            .then(this.handleImportPartialComplete.bind(this))
            .catch(this.handleImportPartialFailed.bind(this));
        }
      }
    }
```

Joonis 8. Import andmepäringute koostamine

Iga päringu vastuse saamisel uuendatakse kasutajaliideses kasutajale nähtavaid tulemusi protessi külgemisest. Ühtlasi toetab päringute tükeldamine veahalduse paremat toimimist. Teenuse kättesaadavust kontrollitakse maksimaalselt iga paari sekundi tagant ning ületamatu vea tekkimisel on võimalik kasutajale kiiresti vastav teave kuvada.

Tulenevalt imporditavate andmete omapärast ning faktist, et xlsx failist sisselugemisel saadakse vaid lahtrisse sisestatud tekstiväärtus ning puudub otsene seos andmebaasis olemasoleva objektiga, näiteks unikaalse identifikaatori näol, tuleb vasteid otsida lahtrisse sisestatud teksti või numbrilise väärtuse põhjal. Selle saavutamiseks otsitakse esmalt unikaalsed sisestatud väärtused salvestatavate tooteandmete hulgast. Joonisel 9 on näidatud unikaalsete väärtuste eraldamine, et hiljem nende põhjal otsida andmebaasitabelitest olemasolevaid andmeobjekte.

```
const aUniqueUnitSymbols = [...new Set([...aExportedProducts.map(o => o.BaseUnit),
                    ...aExportedProducts.map(o => o.ProcurementUnit), ...aExportedProducts
                    .map(o => o.SalesUnit)
                ])];
const aUniqueTypeNames = [...new Set(aExportedProducts.map(o => o.Type))];
const aUniqueStatusNames = [...new Set(aExportedProducts.map(o => o.Status))];
```

Joonis 9. Unikaalsete väärtuste leidmine importimisel

Unikaalste väärtuste põhjal otsitakse vastavatest andmetabelitest kirjeid. Näide andmekirjete otsimisest on näidatud joonisel 10.

```
aUniqueUnitSymbols.length ?utils.dbGenericView.read(db, 'Unit', utils.parseQueryServiceOptions({
                    filter: `Symbol in [${aUniqueUnitSymbols.map(sUnitName =>
`'${sUnitName}'`)}]`
                    })) : [],
aUniqueTypeNames.length ?utils.dbGenericView.read(db, 'Type', utils.parseQueryServiceOptions({
                filter: `Name in [${aUniqueTypeNames.map(sTypeName => `'${sTypeName}'`)}] and
                Entity eq 'Product'`
                })) : [],
aUniqueStatusNames.length ?utils.dbGenericView.read(db, 'Status',
utils.parseQueryServiceOptions({
                filter: `Name in [${aUniqueStatusNames.map(sStatusName => `'${sStatusName}'`)}]
                and Entity eq 'Material'`
                })) : [],
```

Joonis 10. Andmekirjete otsimine andmebaasist unikaalsete väärtuste põhjal.

Järgnevates etappides otsitakse sisestatud tooteandmete väärtustele vastavate andmeobjektide tabelitest leitud olemasolevate objektide unikaalsed identifikaatorid ning seeläbi luuakse seosed varem salvestatud ja import protsessi käigus loodavate toodete jaoks. Kirjeldatud lähenemise eesmärgiks on veahalduse lihtsustamine ja üleüldine funktsionaalsuse optimeerimine.

Veahalduse osas võimaldab varajane seoste otsimine välistada koheselt sisestavad tooted, mille mõnele nõutud väljale ei leitud vastet. Tagatakse olukord, kus andembaasi üritatakse sisestada vaid sobivaid andmekirjeid.

Andmekirjete valideerimisel liigutakse järk-järgult spetsiifilisemate kriteeriumite suunas. Lähtudes peatükis 2.2.3 kirjeldatud mitmekihilise veahalduse implementeerimise headest tavadest veendutakse esmajärgus protsessiks vajalikes nõuetes. Vajalikud teenused peavad olema kättesaadavad. Juhul kui ühendust ei õnnestu luua, kuvatakse kasutajale veateade ja funktsionaalsus peatub. Ülejäänud süsteemi töö jätkub. Äriliste nõuete täitmise ehk tooteandmete nõuetele vastavuse kohta on loodud eraldi kontrollid, mille mittetäitmisel jäetakse kogu konkreetne toode edasisest protsessist kõrvale ning märgitakse vigaseks. Seejärel võetakse valideerimisele järgmine toode ning kogu protsess jätkub. Äriliste nõuete kontrollimisel ei peatata kogu protsessi.

Juhul kui tootekirje valideerimine möödub edukalt, liigutakse edasi salvestamise protsessi juurde. Andmete andmebaasi salvestamiseks kasutatakse lähtuvalt olukorrast kas CREATE või UPDATE päringuid. Andmete sobivusest tulenevalt ei tohiks peale valideermist salvestamisel vigu tekkida. Siiski säilib võimalus, et andmebaasi enda töös on tekkinud tõrkeid ning sel juhul salvestumine ebaõnnestub. Taolises olukorras protsess peatub ning kasutajale kuvatakse veateade.

Valideerimise ja salvestamise tulemusena tagastatakse andmeobjekt, kus on välja toodud salvestatud tooted, vigased tooted, vigade põhjused, vea tekkekohad jne. Tagastatav objekt on nähav joonisel 11.

```
return {
        aUpdatedProducts: aUpdatedProducts,
        aCreatedProducts: aCreatedProducts,
        aInvalidProducts: aInvalidProducts,
        aCreatedLinkedProducts: aCreatedLinkedProducts,
        aUpdatedLinkedProducts: aUpdatedLinkedProducts,
        aInvalidLinkedProducts: aInvalidLinkedProducts,
        aFaultyColumns: [...new Set(aFaultyColumns)],
        aFaultyUnits: [...new Set(aFaultyUnits)],
        oFaultyRows: oFaultyRows,
        aFaultyLinkedProductColumns: [...new Set(aFaultyLinkedProductColumns)],
        aFaultyLinkedProductUnits: [...new Set(aFaultyLinkedProductUnits)],
        oFaultyLinkedProductRows: oFaultyLinkedProductRows
    };
```

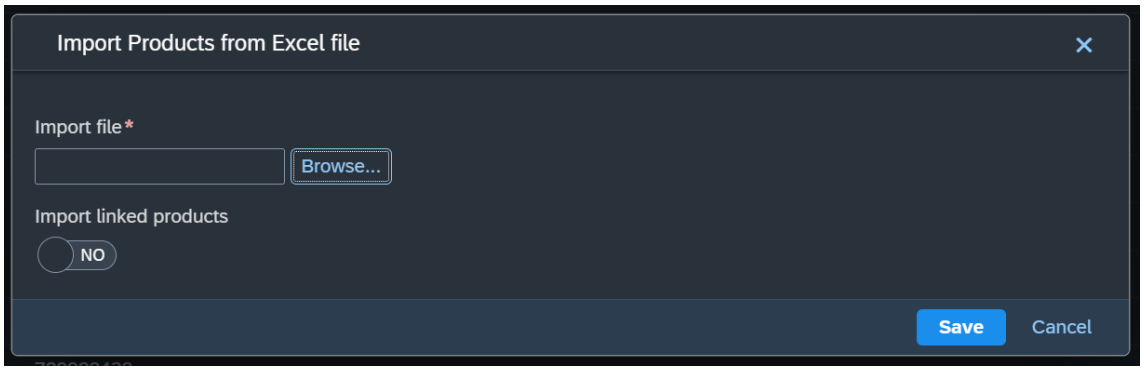
Joonis 11. Import protsessi *backend* funktsionaalsuse tulemina tagastatav andmeobjekt

Saadava vastuse põhjal kuvatakse kasutajale ülevaade protsessi tulemustest. Kuvatava vastuse kasutajaliidese näited on toodud peatükis 4.2.3.

4.2.3 Kasutajaliidese arendamine

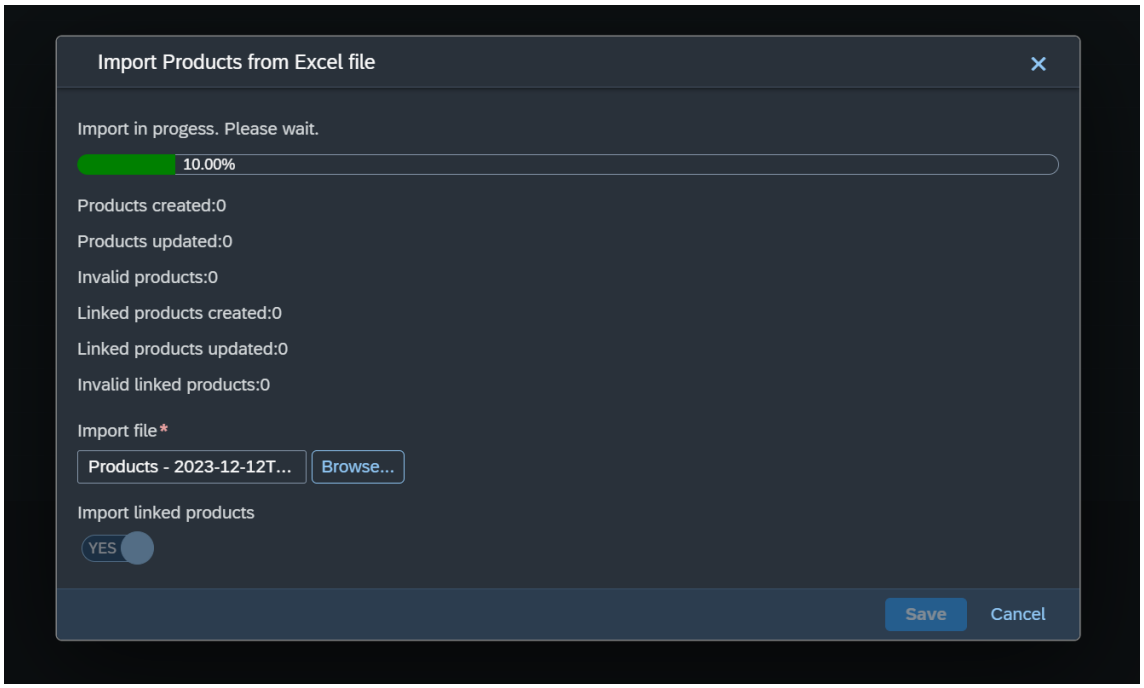
Eksportimise protsessi kasutajaliidese loomine piirdus ühe nupu lisamisega kasutajaliidesesse. Importimise jaoks loodi eraldi dialoog faili valimiseks ning protsessi kulgemise ja tulemuste ülevaate andmiseks.

Esmalt lõi autor dialoogiakna faili valimiseks. Dialoogiaknas peab olema võimalik valida fail importimiseks ning teha valik, kas uuendada või sisestada seotud toodete seosed või mitte. Joonisel 12 on näha arendatud dialoogiakent. Faili valiku nuppu vajutades avaneb File Explorer, kust kasutaja saab valida soovitud faili. Seotud toodete *switch* annab kasutajale võimaluse tooteid omavahel importimisel siduda või seosed kõrvale jätta.



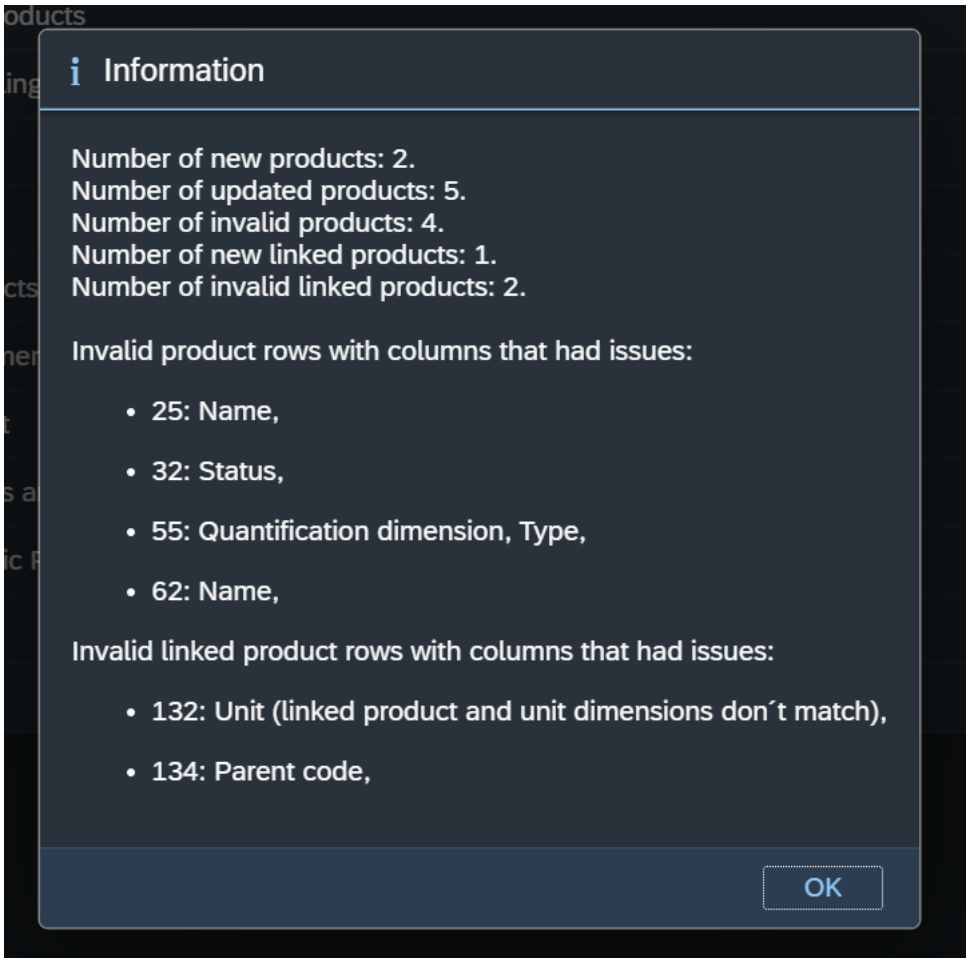
Joonis 12. Dialoogiaken import faili valimiseks

Tulenevalt nõudest, et kasutajal peab olema pidev ülevaade importimise protsessi käigust, koostas autor veel ühe dialoogiakna, mis kuvatakse peale faili valiku tegemist ning salvesta nupu vajutamist.



Joonis 13. Import protsessi ülevaate dialoogiaken

Joonisel 13 on pilt dialoogiaknast, mis kuvatakse kasutajale impordiprotsessi käivitamisel. Välja on toodud sisestatud, uuendatud ja vigaste toodete kogused. Lisaks sellele on kuvatud tooteseoste sisestamise tulemused. Dialoogi ülemises osas on näidatud protsessi progressi, mis pidevalt protsessi käigus uueneb. Töö lõppedes kuvatakse kasutajale ülevaade tulemustest. Dialoogiaken tulemuste kuvamisest on nähatav joonisel 14.



Joonis 14. Dialoogiaken import protsessi tulemuste kuvamiseks

Tulemuste dialoogiaknas on välja toodud sisestatud, uuendatud ja vigaste toodete kogused ning ühtlasi ka toodete omavaheliste seoste sisestamise tulemuste kogused. Lisaks sellele on kuvatud tooteandmete vigade tekkepõhjused reapõhiselt. Samuti on välja toodud tooteseoste vigade tekkepõhjused. Numbriliselt on kirjeldatud vigase toote reanumber MS Excel failis ning tekkepõhjusena on välja toodud MS Excel faili dokumendifaili tulba nimi, kus viga esines. Sel moel on kasutajal selge ülevaade importimisprotsessi tulemustest ja tekkinud vigadest ning nende tekkepõhjustest. Vigaste andmete parandamiseks on viited kuvatud.

4.3 Testimine

Valminud lahendust testiti manuaalselt. Praeguses ettevõtte arengufaasis puudub juurutatud automaattestide lahendus ning seetõttu ei tehtud ka töö käigus arendatud funktsionaalsusele automaatteste. Eksport funktsionaalsuse testimisel lähtuti põhimõttest,

et andmed peavad saama väljastatud samal kujul nagu nad on infosüsteemis nähtavad. Manuaalsel eksportimise testimisel kontrolliti visuaalsel vaatlusel andmete samasust. Testimiseks kasutati erinevaid ettevõtte infosüsteemis paiknevaid tooteandmeid. Lisaks sai lahendus kontrollitud ka ebatavaliste andmetega, näiteks *null* väärtused, ebaregulaarselt esinevad pikad sõnad, erinevad andmetüübid: tekst, arvud, kuupäevad. Testimist viisid läbi töö autor, *product owner* ja teised arendajad. Töö käigus leitud vead parandati ja lõpptulemusena veenduti andmete õigsuses.

Konkreetse töö raames piirduti importimise lahenduse testimisel manuaalse testimisega. Sarnaselt eksport lahenduse testimisele, kasutati ka import funktsionaalsuse kontrollimiseks ettevõtte infosüsteemis paiknevaid tooteandmeid, mis esmalt eksporditi ja seejärel testimis eesmärkidel imporditi. Samuti loodi fiktiivseid testandmeid, mis sisaldasid ka ebakorrapäraseid ja vigadega andmeid, näiteks *null* väärtused, pikad sõnad, erinevad andmetüübid: tekst, arvud, kuupäevad. Väga suur rõhuasetus testimisel langes andmeseoste salvestamise korrektsusele. Näiteks tooteklassifikaatorite lisamisel pidi olema võimalik lisada vaid neid klassifikaatoreid, mis ka päriselt eelnevalt süsteemis olemas on. Loodud funktsionaalsust testisid töö autor, *product owner* ja teised arendajad. Korduva testimise käigus leitud vead parandati ja lõpptulemusena hinnati funktsionaalsus korrektselt töötavaks.

5 Tulemused ja analüüs

Järgnevas peatükis antakse ülevaade lõputöö raames valminud lahendusest, nõuetele vastavusest ning võimalikest edasiarendustest.

5.1 Tulemused

Töö käigus valmis ettevõtte nõuetele ja soovidele vastav lahendus koos kasutajaliidesega, mis võimaldab klientidel tooteandmeid ettevõtte infosüsteemi sisestada ja vajadusel uuendada. Samuti on kasutajatel võimalik tooteandmeid infosüsteemist eksportida xlsx tüüpi faili. Loodud lahendus on kättesaadav ettevõtte poolt pakutava rakenduse *live* süsteemis, esialgu küll *Beta* versioonis, kuid siiski klientidele kättesaadav.

Kolmandas peatükis seatud nõuded said täidetud ning esialgu püstitatud skoobile on arendatud lahendus vastav. Töö käigus tekkinud edasiarenduse vajalikkused kaardistati ning ülevaade neist on nähtav peatükis 5.3.

Arendusprotsessi käigus paranes autori oskus implementeerida kolmanda osapoole lahendusi, suuri andmehulki hõlmavate protsesside optimeerimise võimekus, üleüldine programmeerimisoskus ning veahaldusstruktuuri ehitamise oskus. Kõigi arendatud oskuste näol on tegemist otseste vajadustega edasiste tööülesannete lahendamisel.

5.2 Analüüs

Töö peamiseks eesmärgiks oli muuta kasutajale toodete infosüsteemi sisestamine ja nende massiline muutmine mugavamaks ja kiiremaks. Motivaatorina xlsx failist andmete lugemise võimekuse arendamiseks oli suuresti olemasolevate ja potentsiaalsete klientide suur varasem MS Exceli kasutusmaht. Varasemalt toimus osa tootehaldusega seonduvaid protsesse xlxs failides ning seeläbi oli üks võimalikest lahendustest pakkuda kasutajatele võimalikult tuttavat ja mugavat lahendust, et kolida kasutusel oleva tehnoloogia pealt ümber uude infosüsteemi.

ERP süsteemide efektiivsuse alustaladeks ongi sageli korduvkasutatavad andmed ning seda peamiselt põhiandmete (ingl. k *master data*) näol. Sama efekt väljendub ka antud töö puhul loodud funktsionaalsuse käigus. Tooteandmed sisestatakse süsteemi ja üldjuhul neid lihtsalt kasutatakse edasiste protsesside käigus. Vajadusel on ka olemas võimalus teha andmetes massiline muudatus, näiteks olukorras, kus esmasel importimisel tehti mõni inimtekkeline eksimus ning hilisemalt on vaja mõni konkreetne väärtus kõikidel toodetel ära muuta.

Arendatud funktsionaalsus võimaldab kasutajal süsteemi sisestada kõik soovitud tooted korraga. Varasemalt pidi kasutaja ettevõtte infosüsteemis soovitud tooted ükshaaval loom. Massilise sisestamise peamine eelis on ajaline kokkuhoid.

Toodete eksport lahenduse üks eeliseid on ka tooteversioonide salvestamine. Näitena on kasutajal võimalik eksportida kõik tooted iga teatud ajaperioodi tagant ning seeläbi omada ülevaadet tooteandmete muudatuste üle. Taolise lähenemise plussina on kindlasti ka MS Excelis olemasoleva võimekuse kasutamine. Tooteandmete põhjal saab koostada kokkuvõtteid, graafikuid jne.

5.3 Edasised võimalused

Arenduse käigus valmis esialgsetele nõuetele vastav lahendus, mis aitab täita klientide peamised vajadused. Edasiste arenduste käigus oleks võimalik kasutajakogemust mugavamaks muuta ning lahendust optimeerida. Lisaks tuleks lahendus katta automaattestidega töökindluse tagamiseks.

Kasutajakogemuse mugavamaks muutmise käigus oleks variant täiendada eksportimist osalise toodete loetelu väljastamise võimekusega. Praegune lahendus väljastab eksportimisel kõik kasutajale nähtavad tooted, kuid ühe võimalusena võiks kasutaja saada valida, milliseid tooteid ta soovib eksportida. Näiteks mõne kindla omaduse alusel filtreeritud toodete väljastamine.

Importimise protsessi kasutajamugavuse täiendamisena oleks hea variant parendada protsessi tulemuste kuvamist. Kasutajal võiks olla parem ülevaade tekkinud vigade põhjustest. Hetkel kuvatakse kasutajale tekkinud vigade asukoht MS Excel failis, kuid vea tekkepõhjust üldjuhul ei kuvata. Peamiselt oleks sellisest lahendusest abi relatsioonilise andmebaasiseoste loomisel tekkivate vigade kuvamisel.

Lahenduse optimeerimise arendusena võiks katsetada veelkord erinevaid lahendusi andmebaasipäringute efektiivsuse tõstmisel. Hetkel võtab suure hulga tooteandmete sisestamine või uuendamine küllaltki kaua aega ning seda peamiselt tulenevalt andmebaasistruktuuri omapäradest ja päringute ajakulust. Täpsemate murekohtade selgitamiseks tuleks läbi viia eraldi analüüs ning katsetada uute lahenduste efektiivsust.

Testimise osas tuleks rakendus katta automaattestidega. Töö käigus testiti rakendust manuaalselt ning kuigi võimalikult suur hulk äärmusjuhtumeid (ingl. k *edge case*) sai kaetud ja lahenduse üldist töökindlust testitud, on peamine oht edasiste andmebaasi muudatustega rakenduse töökindluse häirimine. Andmebaasitabelite muutmise käigus tekib oht, et ei arvestada lahenduse ülesehitusega ning eeldatavate andmestruktuuri ja tegelikkuse vahel tekib erinevus.

6 Kokkuvõte

Töö eesmärgiks oli luua funktsionaalsus tooteandmete importimiseks xlsx tüüpi failist SAP HANA andmebaasisüsteemi ning ühtlasi võimekust eksportida andmebaasisüsteemist xlsx faili, eesmärgiga muuta ettevõtte klientidele tooteandmete süsteemi sisestamine ja massiline muutmine mugavamaks ja kiiremaks.

Planeerimisfaasis tutvus autor andmekvaliteedi ja mitmekihilise veahalduse heade tavadega ning kirjeldas kasutatavate süsteemide ja rakenduste olemust. Samuti kaardistas autor koostöös *product owneriga* lahenduse nõuete vajaduse lähtuvalt klientide soovidest ja nõuetest.

Failide sisselugemiseks ja genereerimiseks analüüsis autor erinevaid failigeneraatoreid, mis suudavad xlsx tüübi faile genereerida ja neis sisalduvat infot lugeda. Sõltuvalt nõuetest ja pakutavatest lahendustest tehti valik failigeneraatori kasutamise osas.

Töö teostuse osas selgitas autor arenduse tööprotsessi ja selgitas lahenduse struktuuri, kirjeldas kasutajaliideses tehtud muudatusi ning tutvustas lahenduse testimisega seonduvat.

Viimaks kirjeldas autor töö tulemusi, analüüsis loodud funktsionaalsust ning tutvustas võimalikke edasiarendusi.

Töö käigus valmis funktsionaalsus tooteandmete importimiseks ja eksportimiseks SAP HANA andmebaasisüsteemist xlsx tüüpi andmefaili. Valminud lahendus jõudis *Beta* versioonis rakenduse *live* keskkonda.

Kasutatud kirjandus

1. N. Jain, T. Bagga, and A. Tripathi, “I-ERP Intelligent System Modelling and Interfacing : Excel with SAP HANA,” 2022 *International Conference on Computing, Communication, and Intelligent Systems (ICCCIS),* Nov. 2022, doi: https://doi.org/10.1109/icccis56430.2022.10037607. [Kasutatud november 2023].

2. J. C. Wortmann, “Evolution of ERP Systems,” *Strategic Management of the Manufacturing Value Chain*, pp. 11–23, 1998, doi: https://doi.org/10.1007/978-0-387-35321-0_2. [Kasutatud november 2023].

3. "ERP Modules: Main Features, Functionality, and Workflows," [Võrgumaterjal]. Saadaval: https://existek.com/blog/erp-modulesmain-features-functionality-and-workflows/. [Kasutatud november 2023].

4. "Top 5 ERP System Trends in 2020 to help plan for 2021," [Võrgumaterjal]. Saadaval: https://xcelpros.com/top-5-erpsystem-trends-in-2020-to-help-plan-for-2021/. [Kasutatud november 2023].

5. Haun,J., Hickman, C., Loden, D., Wells,R., Implementing SAP HANA. Boston: Galileo Press, 2013.

6. F. Färber, S. K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner, “SAP HANA database,” ACM SIGMOD Record, vol. 40, no. 4, pp. 45–51, Jan. 2012, doi: https://doi.org/10.1145/2094114.2094126. [Kasutatud november 2023].

7. Franz Färber et al., “The SAP HANA Database - An Architecture Overview,” *IEEE Data(base) Engineering Bulletin*, vol. 35, no. 1, pp. 28–33, Jan. 2012, url: http://sites.computer.org/debull/A12mar/hana.pdf. [Kasutatud november 2023].

8. "Data Validation: What is it, Importance, Types, Pros & Cons" [Võrgumaterjal]. Saadaval: https://www.questionpro.com/blog/data-validation/ [Kasutatud november 2023].

9. M. D. Zio et al., ‘Methodology for data validation 1.0’, 2016.

10. R. Lemmik, K. Karjust, and T. Otto, “Fault tolerance in integration interfaces of business software // International Journal Of Scientific Knowledge (Computing and Information Technology)”, *International Journal Of Scientific Knowledge (Computing and Information Technology) IJSK*, vol. 5, pp. 35–43, 2014, url: https://www.etis.ee/Portal/Publications/Display/508dad92-a1ab-44e6-b7e4-aaa35faa5669. [Kasutatud november 2023].

11. Y. Li, Y. Hu, L. Feng, and W. Yang, “Design and implementation of excel massive data intelligent import system,” Jul. 2010, doi: https://doi.org/10.1109/iccsit.2010.5564593. [Kasutatud detsember 2023].

12. "Power Query" [Võrgumaterjal]. Saadaval: https://learn.microsoft.com/en-us/power-query/power-query-what-is-power-query. [Kasutatud detsember 2023].

13. "Overview of XML in Excel" [Võrgumaterjal]. Saadaval: https://support.microsoft.com/en-au/office/overview-of-xml-in-excel-f11faa7e-63ae-4166-b3ac-c9e9752a7d80. [Kasutatud detsember 2023].

14. "SheetJs" [Võrgumaterjal]. Saadaval: https://sheetjs.com/. [Kasutatud november 2023].

15. "Spreadsheet" [Võrgumaterjal]. Saadaval: https://sapui5.hana.ondemand.com/sdk/#/api/sap.ui.export.Spreadsheet. [Kasutatud november 2023].

16. J. Nielsen, Usability Engineering, Morgan Kaufmann, 1993.

17. P. Meso and R. Jain, “Agile Software Development: Adaptive Systems Principles and Best Practices,” *Information Systems Management*, vol. 23, no. 3, pp. 19–30, Jun. 2006, doi: https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93704.3. [Kasutatud detsember 2023].

18. Visser, J. and Rigal, S. and Wijnholds, G., Building Software Teams: Ten Best Practices for Effective Software Development, O'Reilly Media, Incorporated 2017.

19. "FileUploader" [Võrgumaterjal]. Saadaval: https://sapui5.hana.ondemand.com/#/entity/sap.ui.unified.FileUploader. [Kasutatud detsember 2023].

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

Mina, Roomet Oja

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Massiline toodete sisestamine ja uuendamine SAP HANA andmebaasisüsteemis, kasutades MS Excelis olevaid tooteandmeid", mille juhendajad on Rivo Lemmik ja Jakob Jõgi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

03.01.2024

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Näide genereeritud xlsx dokumendist

Leitav: https://drive.google.com/drive/folders/1dK7dg0F4E_x9-7Fdhftuw0L__WyacXRJ?usp=drive_link