TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mart Potter, 176854IAAM

ADVOKAADIBÜROO SORAINEN AS PÄRANDSÜSTEEMI KAASAJASTAMINE

Magistritöö

Juhendaja: Paul Leis
Dotsent

Tallinn 2019

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: Mart Potter

14.05.2019

Annotatsioon

Diplomitöö eesmärk on kirjeldada parimad praktikad ning pidepunktid, millest lähtudes oleks täidetud eelkõige andmete terviklus ja käideldavus Advokaadibüroo Sorainen AS vaates uue platvormi evitamisel. Uue platvormi valis autor välja ettevõttele bakalaureusetöö raames. Diplomitöös kirjeldatakse pärandsüsteemi moderniseerimist uue platvormi juurutamise näol nii, et andmete terviklus ja käideldavus oleks tagatud. Töö käigus kirjeldas autor olemasolevat arhitektuuri ning selle moderniseerimisega seotud väljakutseid keskendudes olemasolevate liidestuste üleviimisele ning vaadates lähemalt pärandsüsteemi ja uue platvormi vahelise infovahetust. Diplomitöö tulemusena kirjeldas autor andmete migreerimise strateegia ja integratsioonide migratsiooniplaani. Ühtlasi analüüsis autor andmete käideldavuse ja tervikluse tagamise võimalust ning arendas välja ettevõtte jaoks vajaliku automaattesti prototüübi millega saab kontrollida kas andmed on kadudeta üle ja millised on erinevused. Prototüüp võimaldab ka andmeid automaatselt parandada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 51 leheküljel, 4 peatükki, 33 joonist, 7 tabelit.

Abstract

Modernizing legacy system for Law Firm Sorainen

The purpose of this diploma thesis is to explain the best practices and point to fulfil the data integrity and availability for Law Firm Sorainen during the implantation of internal platform. New platform was selected as a part of authors' bachelor's theses. Current theses describes the solutions how to maintain data integrity and availability while modernizing of the legacy platform. Author gave an overview of the current architecture and the challenges that are related to migration of the data and integrations of the legacy system. As a result of the theses author developed as automated test to validate if data in the legacy system matches the data of the new system.

Author found out that one needs to firstly know what exactly is going to be migrated – know your data. In practice it meant that author created a table to describe the current database structure and then evaluated its content with the team – what is the part of that that needs to be migrated. Author found out that only third of the database tables are worth to keep, but from the other hand 60% of the connections and fields are needed. Last but not least, 76% of the data is needed to be transferred. It means that most relevant tables contains many times more data than others.

To maintain data integrity author found out that regional settings needs to match in source and target database as well as specific data formats need to be converted before they can be transmitted to the target database. It means that for instance if there is no date recorded in legacy system it looks as empty string but in Microsoft SQL it is "1900-01-01 00:00:00.000". The same applies to comparing Boolean values.

Situation was even more complicated as the source database is proprietary but target database is relational. It means that source and target database entries cannot be directly compared so author created an algorithm to convert both database entries to JSON string and the compare the hash codes of both string. If the hashes do not match then then algorithm allows to fix source target database entries automatically.

Author concludes that developed automated tests are efficient to find and fix data integrity flaws. The outcome of the thesis was only a prototype of the automated test and it takes additional time to set it up for all the categories, fields and connection. Author will start

using that by creating new tests for most critical categories where business risk is higher. In retrospect author agrees that such a validation procedure and algorithm should have been on the table in the earlier stage of the project.

The thesis is in Estonian and contains 51 pages of text, 4 chapters, 33 figures, 7 tables.

Lühendite ja mõistete sõnastik

Commence	Olemasolev kliendihaldustarkvara, paksu-kliendi arhitektuuriga (*fat client arhitekture)* Windowsi tööjaama klient-server rakendus
Outsystems	Bakalaureuse töö käigus autori poolt välja valitud *low-code* arendusplatvorm
Project „Rachel“	Migratsiooniprojekt mis hõlmab Commence'i funktsionaalsuse ning andmete üleviimist OutSystems platvormile
VBS	*Visual Basic for Script* – Visual Basic skriptikeel Skriptikeel, mille töötas välja Microsoft oma veebibrauserile Internet Explorer. Põhineb programmikeelel Visual Basic, kuid on palju lihtsam ära õppida. Sarnaneb mitmes mõttes JavaScript'ile, võimaldades veebiautoritel lisada veebilehtedele selliseid interaktiivseid juhtimiskäske nagu nupud ja kerimisribad. [7]
VBA	*Visual Basic for Application* – Microsoft Office programmeerimiskeel (makrod)
PHP	*Hypertext Preprocessor,* on platvormist sõltumatu skriptikeel ja kujutab endast alternatiivi Microsoft'i ASP (Active Server Pages) keelele, mis töötab ainult Microsoft Windows NT serveritel. [7]

API

Application Programming Interface, rakendustevahelise suhtluse liidestuse raamistik. Reeglid ja vahendid, mida rakendusprogramm kasutab suhtluseks operatsioonisüsteemiga, andmebaasihalduse süsteemiga või muu juhtprogrammiga, samuti sideprotokolliga; määrab suhtlusvormingud ja sisaldab mitmesuguseid mooduliteeke. [13]

XML

Extensible Markup Language, laiendatav märgistuskeel XML on suvaliste andmete struktureerimiseks mõeldud märgistuskeel, mis loodi eemärgiga võtta see veebis kasutusele HTML'i asemel. Nimelt osutus HTML oma fikseeritud elementide ja atribuutidega paljude ülesannete jaoks liialt piiratuks. [7]

Scrum

tarkvaraarenduse metoodika, mis põhineb iteratiivsusel, sagedasel kontrollimisel ja korrigeerimisel ning inkrementsaadustel, kusjuures nõuded ja lahendused kujundab koostöö mitmekülgsetes rühmades ja pidev tagasiside huvipooltelt [8]

Low-code

OutSystems ise kirjeldab seda kui võimekust disainida ja arendada tarkvara rakendusi kiiresti ja minimaalsel koodi kirjutamisega. See võimaldab oskustega arendajal väärtuse tekitamist kiiremini ja töökindlamalt. Kasutades visuaalselt modelleerimist graafilise kasutajaliidesega, et kirjeldada äriloogikat, kokku panna ja seadistada rakendust. [9]

JavaScript

Netscape'i poolt välja töötatud skriptikeel, mis võimaldab veebiautoritel luua interaktiivseid veebisaite. JavaSctipt

	suudab suhelda HTML-keeles kirjutatud lähtekoodiga ja võimaldab muuta veebilehed dünaamiliseks. [7]
CSS	*Cascading Style Sheets* – „kaskaadlaadistik“. Märgistuskeelse (dokumendi sisse paigutatavate vormingusiltide süsteem - sildid määravad sisu struktuuri ja esitusviisi. Nt HTML, XLM [17]) dokumendi välisilme kirjeldamise formaalkeel [10]
Välisvõti	*foreign key*, relatsioonandmebaasis ühe tabeli väli (veerg) või väljarühm, mis üheselt määrab mingi rea teises tabelis, luues sideme nende kahe tabeli vahel [11]
SQL	*Structured Query Language* - enimkasutatav päringukeel, mida toetavad kõik klient-server keskkonnale projekteeritud relatsioonandmebaasid. [7]
CSV	*Comma Separate Value* - komaeraldusega väärtused porditav failivorming, kus andmebaasikirjed on üksteisest eraldatud komadega. Selles vormingus on iga rida üks kirje, mille väljad on üksteisest komadega eraldatud. Komade järel võib olla suvaline arv tühikuid ja/või tabeldusmärke (tab character), sest neid ignoreeritakse. Kui väli ise sisaldab koma, siis peab kogu väli olema ümbritsetud jutumärkidega. [7]
Microsoft Graph API	*Restful* veebi API mis võimaldab ligipääsu Microsofti pilveteenustele [12]
PowerShell	PowerShell on automatiseerimismootor, mis võimaldab kaugligipääsu erinevatele arvutitele, asünkroonset protsessimist ning tänu WMI (Windows Management

	Instrumentation), COM- ja .NET komponentidele lihtsale kasutusvõimalusele on see väga hõlpsasti laiendatav. [14]
Autentsus	Olemi või ressursi päritolu ehtsus, sealhulgas autorluse vaieldamatus, andmete omadus olla pärit väidetavast allikast, võltsimatus [18]
Salgamatus	Tõendatav eitamatus, mis põhineb süsteemi, teenuse või isiku võimel tõendada - väidetava sündmuse või toimingu asetleidmist - seda tekitanud olemite osalust - sõnumi saatmise või saamise salgamist - lausungi, dokumendi või lepingu kehtivuse eitamist [19]
JSON	Lihtne andmevahetusvorming mis põhineb JavaScripti alamhulgal ning on hõlbus inimlugemiseks ja -kirjutuseks [20]
ITIL	Infotehnoloogia taristu teek mis on IT-teenusehalduse populaarne karkass ja metoodika. ITIL on protsessikeskne, organisatsiooni strateegiaga integreeriv. ITIL sisaldab häid tavasid, meelespeasid, ülesandeid, protseduure [25]

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Soraineni (edaspidi ettevõte) tihedat koostööd tegevad Baltikumi ja Valgevene regionaalsed töörühmad, ühtne tegevus- ja kvaliteedijuhtimissüsteem ning oskusteabe andmebaas on ainulaadsed. Tänu nelja kontori teadmiste ja kogemuste täielikule integreeritusele ja ressursside kombineerimisele saab büroo pakkuda klientidele kiiret, sujuvat ja usaldusväärset õigusabiteenust nii kohalikes kui ka piiriülestes tehingutes. Just nendel põhjustel on Sorainen tavaliselt esimene valik mitte ainult keeruliste kohalike tehingute, vaid ka regionaalsete projektide puhul ning klientide jaoks, kes tegutsevad mitmes Balti riigis või Valgevenes.[1]

2013. aastal kirjutas autor Eesti Infotehnoloogia Kolledžis bakalaureusetöö teemal „Uue arendusplatvormi valimine kliendihalduse infosüsteemi jaoks“. Toona tegi autor töö tulemusena ettevõte juhtkonnale ettepaneku valida uueks kliendihalduse infosüsteemi arendamise platvormiks Outsystems ning tellida infosüsteemi migreerimine tootja käest sisse. Autor leidis, et kuigi visuaalne programmeerimine polnud veel IT arendajate seas suurt populaarsust kogunud, siis antud ettevõtte vajadusi arvestades oli see väga õige ja innovaatiline lahendus. Seda just tingituna asjaolust, et tegemist ei ole IT ettevõttega, vaid advokaadibürooga ning sisemiselt on peamine vajadus realiseerida äripoole nõudmistele vastavalt uusi iteratsioone võimalikult kiiresti ja paindlikult, väikse ressursikuluga ja nii, et tulemus rahuldaks mõlemaid osapooli.

Juhtkond kaalus erinevaid valikuid veel mitme aasta jooksul kui 2017. aasta suvel otsustatigi OutSystem’i kasuks. Tõsi, tootja käest ettevõte migratsiooni sisse ei ostnud vaid võttis endale appi välised arendajad, projektijuhi ja arendusjuhi. Eesmärk oli 2019 lõpuks olemasolev funktsionaalsus uuele platvormile üle viia ning veidi lisaarendusi teha.

Käesolev diplomitöö käsitleb äriloogika, andmemudeli ning andmete üleviimist keskendudes andmemudelile ning andmete tervikluse tagamisele. Diplomitöö eesmärk on paika panna migratsiooni strateegia ning leida moodus kinnitamaks, et pärandsüsteemi (Commence) andmed on muutmata kujul üle viidud. **Autori vastutus kogu projekti juures ongi Commence’i ja OutSystem’i vahelise integratsiooni realiseerimine ning Commence’i väliste integratsiooni üleviimine OutSystem’i platvormile.**

Arhitektuuri ja andmete üleviimise teeb keeruliseks asjaolu, et Commence ei põhine relatsioonilisel andmebaasil, OutSystems aga küll (Microsoft SQL). Ühtlasi kasutab Commence väliste süsteemidega suhtlemiseks vaid CSV eksportimist, OutSystem's on integratsioonid realiseeritud aga ka REST API vahendusel.

Commence ja OutSystem'i vahelisele integratsioonile lisab keerukust asjaolu, et Commence on paksu kliendi arhitektuuriga tarkvara, kus enam kui 300 lokaalset andmebaasi oma muutusi pidevalt tsentraalse serveri poole saadavad. Osa funktsionaalsust on plaanis ühe korraga üle viia, ülejäänud jääb aga teatud ajaks paralleelselt kasutavaks. See aga teeb andmevahetuse eriti kriitiliseks. Seega ei ole tegemist andmete ühekordse ülekandmisega (Ingl „*big bang")* vaid aasta-kahe jooksul andmete mõlemas suunas sünkroniseerimisega, ehk ka andmete kvaliteedi tagamine on pidev protsess, mis lisab veelgi keerukust.

Projekti puhul tuleb kasuks, et ettevõttel on tellija pool ka majasisene kompetents olemas (sertifitseeritud arendaja (autor) ja sertifitseeritud „toote omanik" (IT juht) kes vastutab, et äripoole sisend jõuaks arendajateni. Lisaks on kaasatud arendajad koos arendustiimi juhiga ning eraldi projektijuht.

Diplomitöö keskne osa ongi õige arhitektuuri protsessi paika panek ning Commence ja OutSystem'i vahelise integratsiooni tagamine. Eesmärk on kirjeldad strateegia millest lähtuvalt saaks koostada dokumendi kinnitamaks, et andmed pole migratsiooni käigus kaduma läinud, neid pole muudetud ega täiendatud.

Töö koosneb viiest osast. Esimeses peatükis kirjeldab autor ettevõtet üldiselt ning IT tugiteenuseid. Ühtlasi tutvustab autor migratsiooni projekti, valitud platvormi, oma rolli ettevõttes ja migratsiooni projektis. Teises peatükis kirjeldab autor migratsiooni strateegiat. Kolmandas peatükis vaatab autor lähemalt Commence-i pool eksisteerinud väliseid integratsioone ning nende juurutamist OutSystem'i ning ühtlasi Commence-i ja OutSystemi vahelist integratsiooni. Neljandas peatükis võtab autor luubi alla andmete turvalisuse käsitledes eelkõige käideldavuse ja tervikluse tagamist nii Commence-is kui OutSystem-is ning nende kahe süsteemi vahelises suhtluses. Ühtlasi vaatab autor eriti täpselt tervikluse probleemi andmete ülekandmisel ühest süsteemist teise püüdes leida võimaluse saada kinnitust, et andmete terviklus pole migratsiooni käigus kannatada saanud.

1 Ettevõttest

Sorainen on juhtiv regionaalne äriõigusele spetsialiseerunud advokaadibüroo, mille kontorid asuvad Eestis, Lätis, Leedus ja Valgevenes. Alates büroo asutamisest 1995. aastal on Sorainen nõustanud rahvusvahelisi ja kohalikke kliente kõigis Baltikumi ja Valgevene äriõigust ja maksundust puudutavates küsimustes. [1]

1.1 IT tugiteenused

Nii nagu ettevõte teenindab kliente neljas riigis on ka tugiteenused regionaalsed ning erinevatesse kontorisse on koondunud kompetentsikeskused. IT osakonda kuulub täna 4 inimest Tallinna kontoris, 2 nii Riia kui Vilniuse kontoris. IT üldine kompetentsikeskus paikneb Tallinnas ning kogu grupi IT juhtumine toimub Tallinnas. See on osalt kujunenud nii ajalooliselt (Tallinnas avati esimene büroo), teisalt on Tallinnas ka kõige staažikamad spetsialistid. Enamus IT infrastruktuuri on Tallinna kontoris - suurem osa süsteemide haldusest, administreerimisest ning arendusest toimub just siin. Teiste kontorite IT spetsialistide põhiülesanne on kasutajatoe pakkumine ning teatud üksikute teenuste haldus.

Selleks, et aru saada kuidas on IT ressurss ettevõtte sees jagatud loetleb autor kõigepealt IT tugiteenused:

1) Esimese taseme kasutajatugi
2) Teise taseme kasutajatugi
3) Teenuste haldus
4) Rakenduste haldus
5) Serverite haldus
6) Windows tööjaamade haldus
7) Rakenduste arendus
8) Suhtlus väliste koostööparteritega
9) Sise- ja välisvõrgu haldus
10) IT strateegia väljatöötamine ja juurutamine

11) Lõppkasutaja IT koolituste läbiviimine
12) Lõppkasutaja IT alase teabematerjali koostamine (Wiki) ning koolitamine
13) IT turvahügieeni tagamine serverites ja tööjaamades
14) Rakenduste õiguste haldus

Seega on IT kompetents asutuse üsna laia spektriga ning see ulatub nii kasutajatoest kuni rakenduste halduse ja arenduseni välja, hõlmates ka koolitusmaterjalide koostamist ning strateegi väljatöötamist.

1.2 Rakenduste arendus

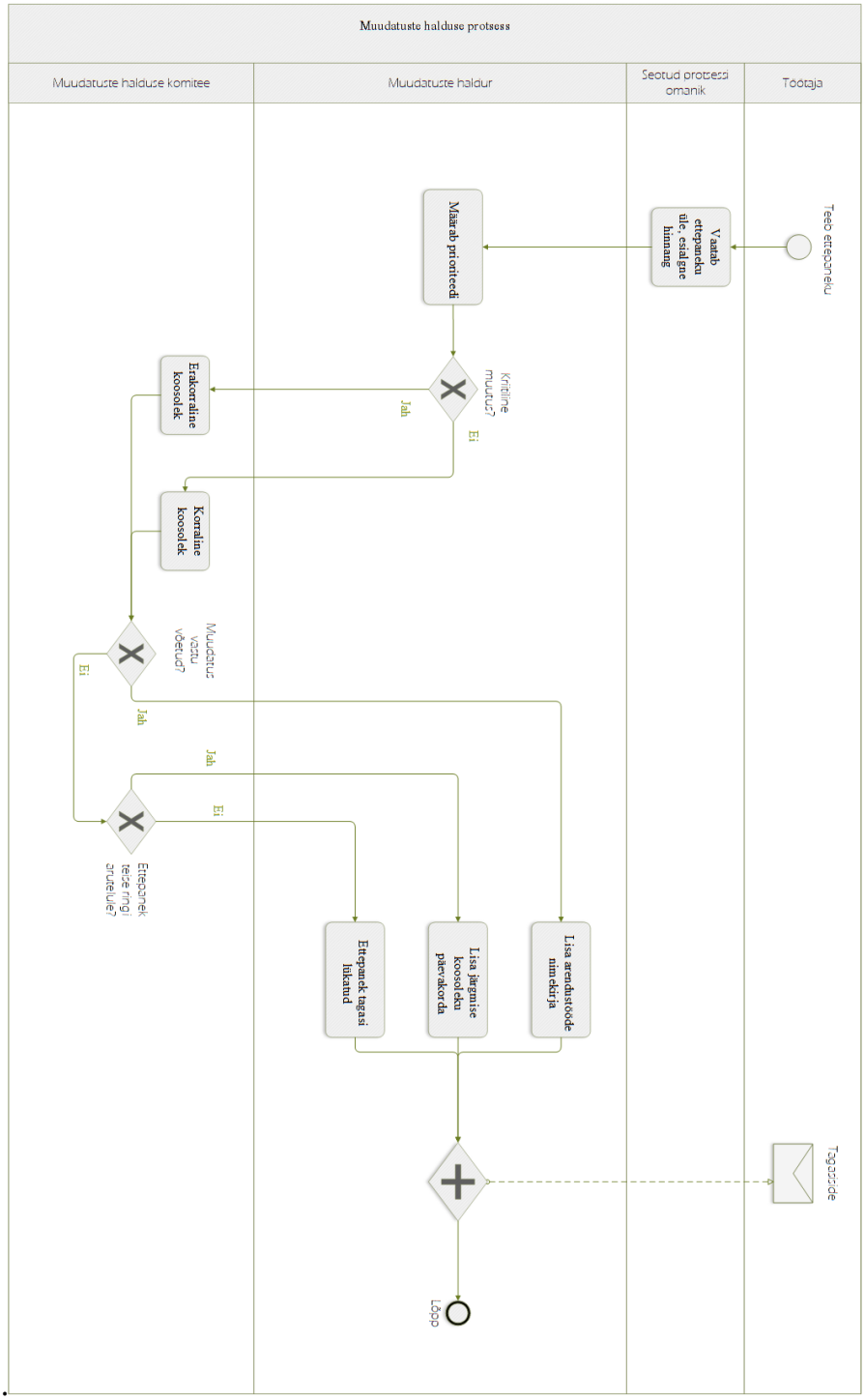
Ettevõtte sees on arenduskompetents enamuses sisemiselt kasutavate tarkvarade jaoks (k.a. Commence ja OutSystems) mis teeb muudatuste tegemise paindlikuks ja kiireks. Seni on arendus pigem *ad-hock* tasemel olnud, ehk tellitud funktsionaalsust arendatakse siis kui selleks on vaba ressurssi.

1.2.1 Muudatuste halduse protsess.

Selleks, et uue platvormi juurutamise käigus saadud tagaside ja ettepanekud läbi vaata ja vajadusel õiges järjekorras töösse võtta kirjeldati ITIL'ist lähtuvalt muudatuste halduse protsessi (Joonis 1 - Muudatuste halduse protsess, autori joonis).

1. Kõik töötajad saavad teha muudatusettepanekuid nii kasutajatoe e-maili teel kui OutSystem'isse integreeritud tagasiside mikroteenuse kaudu.
2. Kõik muudatuste ettepanekud registreeritakse automaatselt Trello töölaual
3. Muudatuste haldur (kes ühtib ka OutSystem'i *scrum* projekti tooteomaniku rolliga) vaatab registreeritud ettepanekud jooksvalt üle, tema puudumisel teeb seda vanemarendaja (autor)
4. Muudatuste halduse komitee koguneb üle nädala (või erandkorras kriitilise muutuse jaoks) kas füüsilise või virtuaalse koosoleku vormis.
5. Koosolekul vaatavad komitee liikmed (kuhu autor vanemarendajana kuulub) kogunenud tagasiside üle. Tehakse lõplik otsus mis läheb arendusse ning milline on prioriteet. Ühtlasi pannakse kirja ettepanekud mis jäävad ootele või lükatakse tagasi
6. Teatud ettepanekud võivad minna teisele ringile – siis kutsutakse muudatusettepaneku tegija järgmisele koosolekule oma ettepanekut kaitsma.

7. Koosoleku järgselt saadab muudatuste haldur kokkuvõtte otsustest ettepanekutega seotud isikutele



Joonis 1 - Muudatuste halduse protsess, autori joonis

1.3 Autori vastutusalad

Autor on ettevõttes töötanud aastast 2008 ja kui alguses oli autor esimese taseme tugi siis täna toetab vajadusel teise taseme kasutajatoega. Autori igapäevatöö põhirõhk on kliendihaldustarkvara rakenduste arendusel (Commence (VBS) ja OutSystem) aga ka Microsoft SharePointil baseeruva siseveebi (.Net) jt veebiteenuste (PHP, WordPress baseeruvad) arendus ja haldus. Regionaalselt on kasutusel Microsoft Windows 10 operatsioonisüsteem ning selleks, et tagada ühtlane tarkvarade seis monitoorib autor seda VBS'iga kirjutatud kliendirakendusega (PHP *front-end rakendus*).

Kontoritarkvarana on ettevõttes kasutusel Micorosft Office 365 ning selle mugavamaks kasutamiseks on autor aastate jooksul kirjutanud erinevaid laiendusi VBAga. Sinna alla kuuluvad näiteks erinevate funktsioonidega kohandatud nupud Wordis, Outlookis ja PowerPointis.

Viimase paari aasta jooksul on autor ennast täiendanud just OutSystem'i arendusplatvormi kasutamise osas ning täna on autori primaarne projekt just Commence funktsionaalsuse ja andmete üleviimine OutSystem'i platvormile. Autor vastutab projektis integratsioonide (eelkõige kahe süsteemi omavahelise infovahetuse) eest ning tegelen ka *front-end* ja *back-end* arendusega.

OutSystem'iga on autor kokku puutunud igapäevasel viimased 2 aastat. Autor on läbinud nädalase arendajate koolituse ning erinevaid täiendõppe mooduleid. 2018 detsembris sooritas autor edukalt esimese astme spetsialisti eksami ja sai *Associate Web Developer* sertifikaadi mis on ära toodud Lisa 1 - Associate Web Developer sertifikaat.

1.4 Projekt "Rachael"

Koodnime "Rachael" alla käib kõik alates Commence'i olemasoleva funktsionaalsuse ning äriloogika kaardistamisest kuni uue lahenduse loomisega OutSystem platvormil koos olemasolevate andmete, äriloogika ning integratsioonide migratsiooniga. Projekt saab läbi kui kogu olemasolev funktsionaalsus on uuele platvormile üle viidud.

Projekti töö käib *Scrum* metoodikaid järgides ning sinna kuulub peale vanemarendajast autori veel ettevõtte poolt nooremarendaja ja toote omanik (autori otsene ülemus). Arendusettevõttest oleme kaasatud 3 arendajat, arendustiimi juhi ja projektijuhi.

Lepiti kokku, et vajadusel on ka kõik välised arendajad meie juures kohal, et oleks operatiivsem omavahel ja äripoolega suhelda. Sprindid on kokkulepitult kahe nädala pikkused, algavad planeerimisega ning lõppevad kokkuvõtte ja tagasivaatega. Mitu korda nädalas tehakse püstijala koosolekuid (*stand-up meeting*) kus arutatakse käesolevat tööd ja võimalikke probleeme.

Projekti plaan sai tehtud ambitsioonikas kuid selles on väiksemate mööndustega siiani kinni peetud. Plaaniga saab lähemalt tutvuda Lisa 2- Projekti plaan.

Kogu projekt on jagatud erinevatakse töödeks ning tööde haldus käib Trello töölaual. Töölaud on jagatud erinevateks veergudeks (planeeritud tööd, hetkel töös, ülevaatamisel, valmis jne). Iga sprindi alguses valivad arendajad endale planeeritud tööde seast vastavalt koormusele sobiva koguse ülesandeid (kaarte) ning hindavad kui palju selle kaardi tegemiseks aega läheb. Kaardid liigutakse „sprindi“ veergu. Kui tööga alustakse liigutakse see tulpa „tegemisel“.Iga kaardiga peale kulunud tegeliku aega mõõdetakse Toggl abiga (Toggl ja Trello töölaud on omavahel liidestatud) – sprindi lõppedes võrreldakse kas prognoositud ja tegelik aeg klappisid. Kui töö saab valmis siis liigutakse see „ülevaatamisel“ tulpa ning lisatakse ülevaataja kes saab siis töö vastu võtta või tagasi lükata.

Lisaks näost-näkku suhtlusele kasutakse erinevaid kommunikatsioonivahendeid (Microsoft Teams, e-maili listid).

Töö käigust kirjutatud kood hoitakse BitBucket’i repositooriumis. Iga kaardi kohta tehakse eraldi haru ning enne haru sulgemist peab teine arendaja tehtud muutused üle vaatama ning kinnitama. Arendatud koodi hoitakse nii arendus, kvaliteedikontrolli- kui *live*-keskkonnas.

1.5 Outsystems

Bakalaureuse töö käigus autori poolt välja valitud Outsystems[1] on Portugali iduettevõttega kes äsja sai ka *Unicorn* liigasse (ületas miljardi euro väärtuse). Tegemist on visuaalse programmeerimise arendusplatvormiga.

[1] https://www.outsystems.com/

Süsteemi ehitatakse moodulite kaupa nii, et teatud funktsionaalsus on samm-sammult kätte saadav ka uues platvormis kuid seni kuni kogu funktsionaalsus pole üle toodud peavad nii vana kui uus süsteem ka omavahel väga tihedalt integreeritud olema. Antud valik tehti seetõttu, et sooviti anda võimalus kasutaja teatud funktsionaalsust paralleelselt nii Commence'is kui OutSystem'i kasutades ära näiteks OutSystem'i mobiilsust (võimalus kirja panna oma tehtud töö ka nutitelefonist) ning seega rohkem tulu teenida.

Andmed, mis olid seni vanas süsteemis tuleb teisendada relatsioonilise andmebaasi jaoks. Andmete lisandumine / muutmine Commence'is peab kajastuma ka OutSystem'is ja vastupidi. Antud protsessi teeb keeruliseks ühelt poolt see, et Commence'i andmebaas ei ole relatsiooniline ning teisalt asjaolu, et optimaalse tulemuse saamiseks on uue andmebaasi arhitektuur olemasolevast erinev.

Commence'is on palju integratsioone ning kõik need tuleb sujuvalt üle tõsta – see on ka autori peamine ülesanne. Täna on projekti meeskond uut süsteemi ehitanud üle pooleteise aasta. Teatud integratsioon e-posti süsteemide (automaatsed teated, kalendrikirjed) töötavad vaid OutSystem'i platvormil ning vanas süsteemis seda funktsionaalsust enam üldse kasutada ei saa (seda infot ei saadetagi üldse Commence'isse tagasi ning integratsioon on peatud).

OutSystemi puhul on tegemist *low-code* arenduskeskkonnaga kus enamus arendust saab teha nii, et koodi kirjutama ei pea. Kui seda tehakse siis kas keerulisemate andmebaasi päringute (*Microsoft SQL*), veebi automaatika (*JavaScript*) või stiilide (*CSS*) jaoks. Lisaks on võimalik kirjutada oma laiendusi *.Net*'is. Ühtlasi saab kasutaja ka teiste OutSystemi arendajate loodud laiendusi.

1.6 Ettevõtte kokkuvõte

Autor leidis, et ettevõttes on piisavalt kompetentsi, et sellises mahus arenduses osaleda kaasates välised partnerid. Autori hinnangul oli mastaapse projekti jaoks kriitiline välja töötada muudatuste halduse protsess mis võimaldab efektiivselt analüüsida kasutajate soove ning neid siis prioritiseerides töösse võtta.

2 Andmete migreerimise strateegia

Autor soovib kindlaks teha, kas Commence'isse algsel kujul lisatud andmed jõudsid integratsioonikihis täpselt samal kujul OutSystems'i andmebaasi ning kas OutSystemsis sisestatud andmed jõudsid muutmata kujul Commence'isse.

Ivan Soto on IVT Network artiklis kirjeldanud mõned andmete migreerimise strateegia väljakutsed [2]

- Selge arusaam mis andmeid soovitakse üle kanda
- Selge arusaam andmete riski mõjust
- Andmete tervikluse tagamine migratsiooni käigus
- Adekvaatse andmete migratsiooni strateegia defineerimine
- Adekvaatse andmete migratsiooni verifitseerimise strateegia defineerimine

2.1 Selge arusaam mis andmeid soovitakse üle kanda

Andmete kaardistamisega alustati projekti algusfaasis ning esimese hooga kirjeldati autori eestvedamisel pärandsüsteemi struktuuri järgi uue süsteemi *AS-IS* struktuur. Siinkohal on väga kasulik asjaolu, et sellel hetkel oli nii üks arendaja kui projekti omanik ettevõtte sisesed.

Selles etapis koostas autor tabeli ning kirjeldas detailselt ära olemasoleva andmebaasi struktuuri. Kirja sai pandud kõik kategooriad ning nende vahelised seosed. Iga kategooria kohta tegi autor eraldi vahelehe, kus kirjeldas just selle kategooria väljad, välisvõtmed ning seosed teiste tabelitega. Tabel 1 - Kategooria "Banktransfer" veerud toob välja ühe näidiskategooria veerud ning andmetüübid. Kui teksti ja numbri tüüpidega on SQL andmebaasis lihtne opereerida, siis valik, mis on oma olemuselt üks-mitmele-seos vajab SQL'is vahetabelit ning viidet välisvõtmele.

Tabel 1 - Kategooria "Banktransfer" veerud, autori tabel

id	Nimi	Staatiline tabel	Tüüp	Kohustuslik	Tähemärke
1	TransactionID		Nimi	Jah	50
2	Amount		Number	Ei	80
3	AmountMatchWith	BankTransfer_ AmountMatchWith	Valik	Ei	0
4	Bank		Tekst	Ei	80
5	Date		Kuupäev	Ei	80
6	Description		Tekst	Ei	30000
7	Invoice marked paid		Jah/Ei	Ei	0
8	N/A to any invoice		Jah/Ei	Ei	0
9	Notes		Tekst	Ei	30000
10	Partial Payment		Jah/Ei	Ei	0
11	Transfer from		Tekst	Ei	80

Lisaks kirjeldas autor välisvõtmed (üks mitmele) ning mitu-mitmesed vahetabelid mis on näitena ära toodud Tabel 2 - Kategooria "Banktransfer" seosed.

Tabel 2 - Kategooria "Banktransfer" seosed, autori tabel

Seos	Kategoriasse	Üks-mitmele	Mitu-mitmele
has	currency	BankTransfer > Currency Identifier	
has	Entities		Entity_hasBankTransfer
has	invoice		Invoice_hasBankTransfer
has	office	BankTransfer > Office Identifier	

Kui *AS-IS* andmemudel oli täielikult kirjeldatud vaatas autor koos toote omanikuga andmestruktuuri üle ja märkis ära need tabeli, veerud ja seosed mida kindlasti üle kanda vaja pole. Näiteks oli pärandsüsteemis arendatud raamatute register mis on tänaseks migreeritud siseveebi ja mille kasutajaliides oli lõppkasutaja jaoks külmutatud, samas olid aegunud andmed endiselt alles.

Sellise välistusmeetodiga sai kõrvaldatud kohe 66% andmetabelitest mis aga ei tähendanud, et vaid 33% andmetest üle tuleks üle kanda, tegelikud numbrid on ikka suuremad nagu ilmestab Tabel 3 - Ületoomist vajavate tabelite, seoste, väljade ning kirjete koondraport See on põhitabelitest on lihtsalt kordades rohkem andmeid kui nendes, mida üle tuua vaja pole.

Tabel 3 - Ületoomist vajavate tabelite, seoste, väljade ning kirjete koondraport. Autori tabel

#	Pärandsüsteemis	Vaja üle tuua	%
Tabelid	116	39	33,62%
Seosed	722	435	60,25%
Vaated	804	612	76,12%
Väljad	1269	723	56,97%
Kirjed	**1 320 245**	**1 006 872**	**76,26%**

Kui oli teada milliseid andmeid on vaja kasutada siis ehitas autor integratsioonikihi mis

a) Sisestab olemasolevad andmed ühekordselt põhi ja vahetabelitesse

b) Lisab Commence'i klientide poolt tehtud muudatused järjekorda (Ingl. *queue*) ning lisab, muudab või kustutud andmeid OutSystems'is

c) Lisab OutSystems'is tehtud muudatused järjekorda ning lisab, muudab või kustutud andmeid Commence'ist.

Loomulikult ei olnud selline ehitus ühekordne ettevõtmine vaid selle täiendamine käib pidevalt edasi nii autori kui teiste arendajate poolt.

2.2 Selge arusaam andmete riski mõjust

Selles etapis arutas autor koos tiimiga millised andmed on ettevõttele kriitilisemad ning milliste osade kaupa funktsionaalsus üle viia. See tähendab, et teatud osad sai ühe korraga migreeritud ning neid enam Commence'is kasutada ei saagi, teine osa on aga paralleelselt kasutatav mõlemas süsteemis. Näiteks tehtud tööde registri (tunnikaart) tugi jäi kättesaadavaks ka vanas süsteemis kuna seal oli ka klientide ja projektide aga ka kogu arvelduse osa. Selleks, et arvet vormistada võib endiselt vaja minna mõnda arverida (tehtud tööd) muuta ning seda on kiirem teha kui vastav funktsionaalsus on endiselt avatud. Samas tekitab see igapäevaselt rohkem andmete sünkroniseerimist.

Ettevõtte kontekstis on **andmete risk** seotud eelkõige tervikluse ja käideldavusega, ehk siis:

1. Tervikluse risk on migratsiooni käigus see, kui
   a) Commence sisestatud andmed ei jõua muutmata kujul OutSystemsisse

   b) OutSystemsis sisestatud andmed ei jõua muutmata kujul Commenceisse

c) Andmed on Commence ja/või OutSystemsi kättesaadavad selleks volitamata isikutele

2. Käideldavuse risk on migratsiooni käigus see, kui

a) Commence sisestatud andmed ei jõua piisavalt kiiresti OutSystemsisse

b) OutSystemis sisestatud andmed ei jõua piisavalt kiiresti Commence'isse

Kui vaata projekti (Lisa 2 - Projekti plaan) veidi täpsemalt siis võib siinkohal välja tuua mõned ajaliselt verstapostid:

1) Aprill 2018

    a. Tunnikaartide lisamine ni Commence'is kui OutSystems'is

2) Juuni 2018

    a. Persoonide haldus nii Commence'is kui OutSystems'is

3) August 2018

    a. Töötaja profiili haldus, koolituste ja kontorist väljasoleku teadete register – seni vaid Commence'is, edaspidi ainult OutSystem'is

4) Märts 2019

    a. Klientite ja projektide haldus nii Commence'is kui OutSystems'is

Selleks, et riske maandada jätsime klienditööga seotud funktsionaalsuse lahti ka Commence'is. Ivan Soto leiab, et kriitilised andmed on need millel on otsene mõju kriitilistele protsessi parameetritele ja kriitilistele kvaliteedi atribuutidele [2]. Ettevõtte jaoks on selleks tunnikaart ehk arve rida. Kui see info on vale või tuleb viitega on ka arve vale ja seda ei saa lubada. Loomulikult on kriitilised ka kliendi ja projekti infoga seotud andmed kuna ilma nendeta ei saa tööd teha, tunde kirja panna ega arvet esitada.

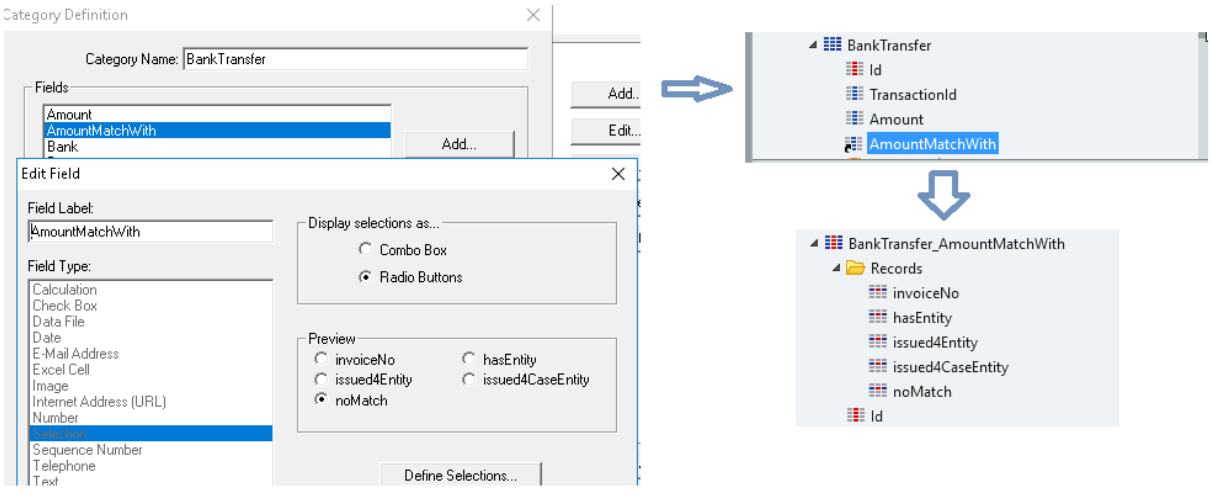
2.3 Arhitektuuri loomine

Lisaks andmetabelite nimekirja valideerimisele on Ivan Soto arvates sama tähtis määratleta andmete iseloom ja struktuur [2]

- Pärandsüsteemi andmete vorming
- Uue süsteemi andmete vorming
- Pärandsüsteemi andmete maht
- Pärandsüsteemi regulatiivsed piirangud
- Pärandsüsteemi andmete säilitamise periood

Selles etapis kasutas autor eelnevalt kaardistatud olemasolevat andmemudelit ja tegi põhi -ja abitabelid õigete tüüpidega OutSystem platvormil.

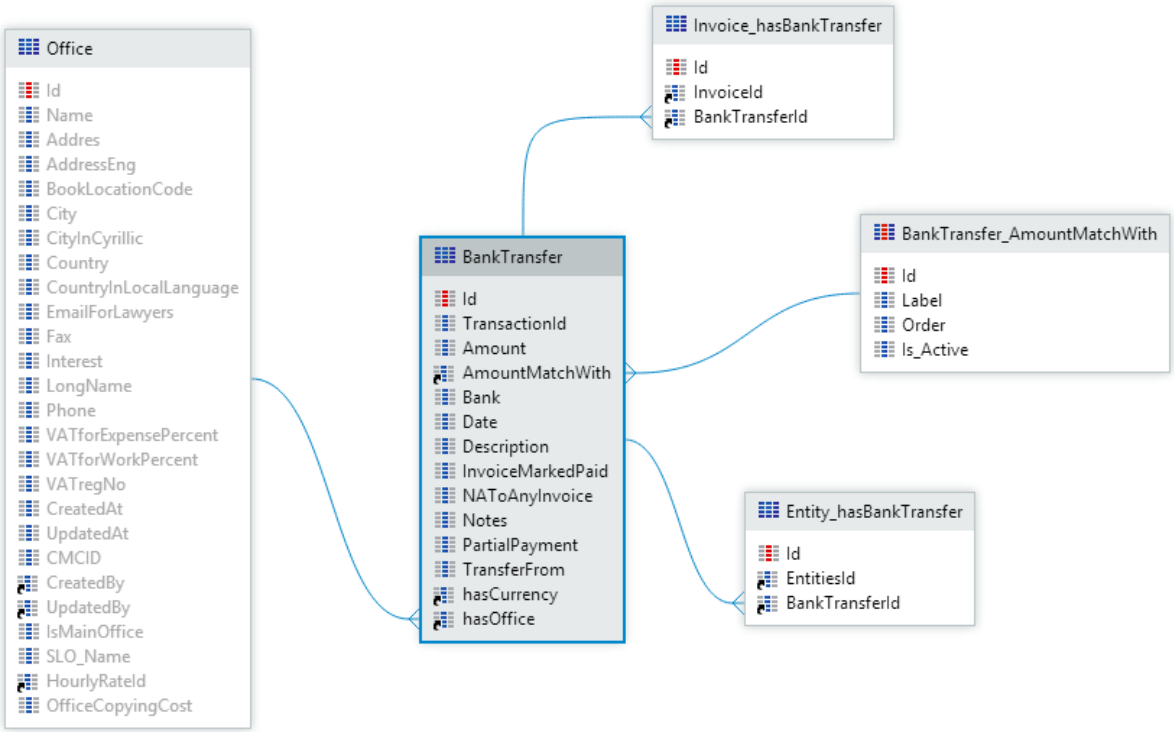
Kui Commence'i kategooriates saab OutSystemsi (SQL'is) teha lihtsalt tabelid ning Commence teksti/numbri väljad on ka Outsystemsis lihtsalt veerud, siis näiteks valikvastuste (*radio button*) realiseerimiseks on vaja juba abitabelit (*static entity)* ja välisvõtit – selle lahenduse realiseerimist illustreerib *Joonis 2 - Raadionupu realiseerimine Commence'is ja Outsystemsis.*



Joonis 2 - Raadionupu realiseerimine Commence'is ja Outsystemsis, autori joonis

Lisaks raadionupu lisatabelile tuleb ka kõik mitu-mitmesed tabelid eraldi SQL tabeliteks luua. Joonis 3 - "Banktransfer" tabeliga seotud olemisuhtediagramm näitel on mitu-mitmesteks tabeliteks *Invoice_hasBanktransfer* ja *Entity_hasBankTransfer*. See tähendab, et ühel ülekandel (Ingl. *Banktransfer*) võib olla seotud mitu arvet ning ühel arvel võib olla seos mitme arvega. Teisalt võib ühel kliendil olla seos mitme ülekandega ning üks ülekanne võib olla seotud mitme kliendiga. Commence'i poole peal täidab need

seosed ära autori kirjutatud VBS import-kood mis töötleb pangaväljavõtte XML dokumenti. Sarnane integratsioon tuleb projekti hilisemas faasis luua ka Outsystem'i poolele või siis vahetada see välja REST API'ga.



Joonis 3 - "Banktransfer" tabeliga seotud olemisuhtediagramm, autori joonis

Andmete säilitamisele pole seni hetkel kitsendusi pandud, seni on ainuke piirang olnud Commence'i andmebaasi võimekus salvestada vaid kuni 500'000 kirjet. Kuna ühes kategoorias hoitakse kõikide tehtud tööde kirjeid siis see piir on ammu ületatud. Lahenduseks on seni olnud paar korda aastas vanade kirjete arhiveerimine teksti faili nii, et ärianalüüsitarkvara arhiivifaili kasutada saaks. OutSystem'is (Microsoft SQL) sellist piirangut pole ning seal pole ka vajadust neid kirjeid arhiveerida. Andmete arhiveerimine Commence pool oli selge piirang kuna aeg-ajalt oli vaja väljavõtteid ka arhiveeritud kirjetest ning see eeldas, et autor pidi kasutaja jaoks arhiivist väljavõtte tegema ja selle loetavale kujule (nt Excel) teisendama.

Regulatiivselt on ette nähtud, et Eestis tuleb arveid säilitada 10 aastat. See on täidetud, sest kõik arved hoitakse alles ning seda ka mitmes erinevas versioon - metaandmetena nii Commence'is kui raamatupidamistarkvaras, koos saadetud kirjaga dokumendihaldussüsteemis ning paberivaba arvearhiivi veebiteenuses.

2.3.1 Kihiline arhitektuur

OutSystem'i platvormi kõige optimaalsem arhitektuur OutSystem'i enda hinnangul 4 kihiline. Tänu selle saab teenusest orienteeritud arhitektuuri disainimise teha lihtsaks. See tagab taaskasutavate mikroteenuste hea isolatsiooni ning võimaluse rakenduse erinevaid kihte korraga arendada. Ühtlasi võimaldab see veaparandusi paigata väikeste tükkide haaval nii, et ei pea kogu rakendus arendus keskkonnast *live*'i saatma. [23]

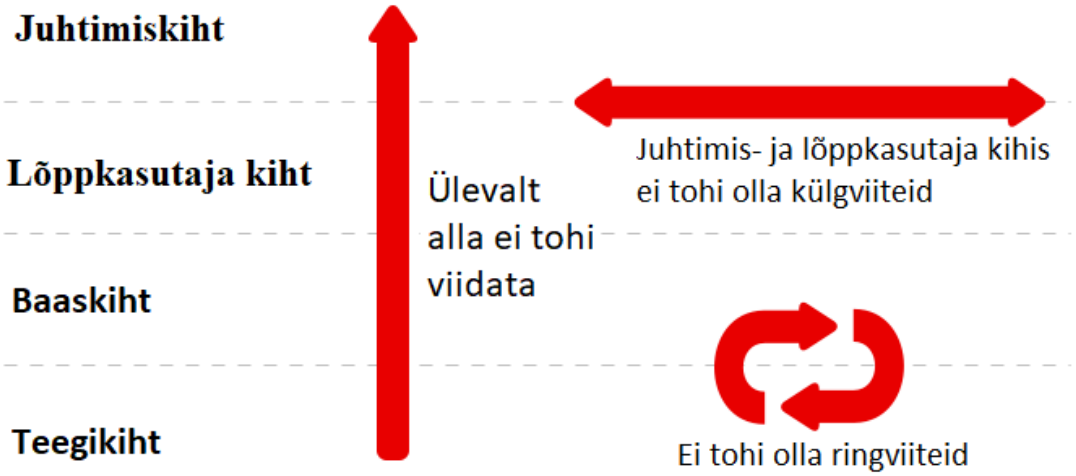
Iga kiht koondab endas erinevaid funktsionaalsusi mida kujutab Tabel 4 - OutSystems kihilise arhitektuuri tabel [23], autori tõlge.

Tabel 4 - OutSystems kihilise arhitektuuri tabel [23], autori tõlge

	**1. Juhtimiskiht** Rakenduste ülene juhtimine	Protsessid, töölauad ja portaali esilehed, erinevatel rakendustel kokku pandud informatsioon mis pakub kasutajakogemust.
	**2. Lõppkasutaja kiht** Kasutaja protsessid	Kasutajaliidesed ja protsessid, baas- ja teegiteenuste taaskasutamine, et kasutajalugusid realiseerida
	**3. Baaskiht** Põhilised äriteenused	Ärikontseptide ümber koondatud teenused, taaskasutatavate tabelite eksport, ärireeglid ja ärikomponendid.

	**4. Teegikiht** Mitteärilised teenused	Äri mittetunnetuslikud teenused mis laiendavad raamistiku enim taaskasutavate komponentiga, kasutajaliides mustrid, liidestused väliste süsteemidega

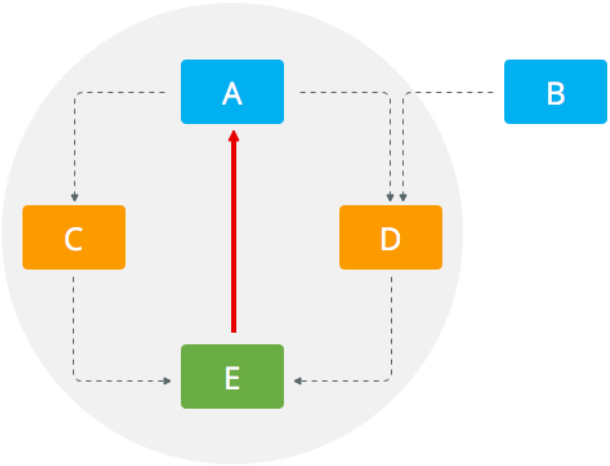
Arendatud moodulid tuleb paigutada õigesse kihti järgides kolme põhireeglit: [24] mida iseloomustab Joonis 4 - Kihilise arhitektuuri põhireeglid



Joonis 4 - Kihilise arhitektuuri põhireeglid , OutSystems joonis [23], autori tõlge

3. Ülevalt alla ei tohi viidata

   Kui selle reegli vastu eksida võib tekkida olukord, kus suvalised kaks moodulit on otseselt või kaudselt ringsõltuvuses. Joonis 5 - Ülevalt alla viitamise näide illustreerib olukorda kus teek E tarbib lõppkasutaja kihist moodulit A – iga elementide paar kõnealuses kobaras on ringsõltuvuses. Näiteks C ja D: C > E > A > D ja vastupidi D > E > A > C. Teine soovimatu tulem on asjaolu, et lõppkasutaja moodul B tarbib õigustatult baaskihi moodulit D ning muutub seega sõltuvaks kogu kobarast [23].



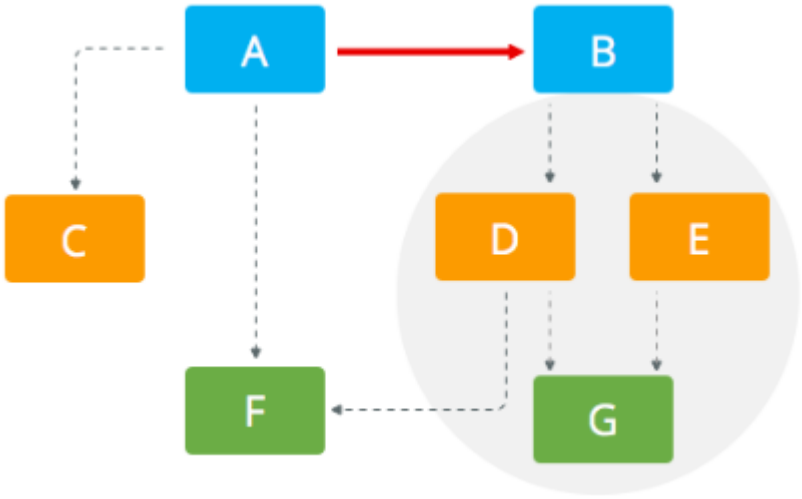
Joonis 5 - Ülevalt alla viitamise näide, OutSystems joonis [23]

4. Juhtimis- ja lõppkasutaja kihis ei tohi olla külgviiteid

Lõppkasutaja- ega juhtimiskiht ei tohiks pakkuda taaskasutavaid teenuseid. See tagab, et need on korrektselt isoleeritud ning neile on lubatud erinevad elutsüklid – erinevad versioonid tingituna asjaolust, et nende arendusega võivad tegelevad erinevad inimesed või need on seotud erinevate töödega[23].

Isolatsioon on kriitiline kuna kõnealused kihid asuvad arhitektuuri tipus. Selline viide võib tuua esile suure koguse kaudseid sõltuvusi madalamates kihtides [23].

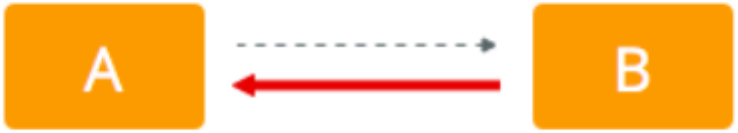
Joonis 6 - Külgviite näide juhtimis- ja lõppkasutajakihis iseloomustab olukorda kus lõppkasutaja kihis olev moodul A tarbib mikroteenust samas moodulis olevast moodulist B (kasvõi minimaalne stiilimise funktsioon). Mitte ainult ei sattunud see nüüd paari mooduliga B, vaid sai ta kaasavaraks ka ebavajalikud seosed moodulitega D, E ja G [23].



Joonis 6 - Külgviite näide juhtimis- ja lõppkasutajakihis [23]

5. Baas- ja teegikihis ei tohi olla ringviiteid

   Kolmas reegel, mida iseloomustab Joonis 7 - Baas- ja teegikihi külgviite näide tähendab seda, et kui moodul A viitab moodulile B siis moodul B ei tohi viidata tagasi moodulile A – vastasel juhul ei saavuta kunagi olukorda, kus kõik viited on ajakohased. Ehk siis kui A tarbib mikroteenust mida pakub B siis B laiendab A'd [23].



Joonis 7 - Baas- ja teegikihi külgviite näide [23]

Projektis „Rachael" tänaseks valminud komponentide neljakihilise arhitektuuriga saab tutvuda Lisa 1 - Commence'i poolt andemete muutmise töövoog.

2.4 Migreerimise strateegia kokkuvõte

Autor leiab, et andmete migratsiooni strateegia on väga tähtis ning sellega alustati projekti alguses kohe õiges suunas, st andmete kaardistamisest ning mahu hinnangust. Hinnates andmeid mida on vaja üle tuua leidis autor, et vaid kolmandik andmetabelitest on endiselt relevantsed samas kui andmetabeli kirjetest on vaja üle tuua kolmveerand. Andmete riski

mõju hindamisel leidis autor, et teatud andmetabelid on kõrgema riskiga millega kaasneb ka nende andmete täpsem monitooring ning tervikluse valideerimine.

3 Integratsioonid

Selles peatükis vaatab autor täpsemalt Commence'i ja Outsystem'i vahelist integratsiooni ning kirjeldab Commence'i olemasolevate integratsioonide üleviimist OutSystem'i platvormile. Lähtuvalt asjaolust, et päris pikalt on kaks süsteemi paralleelselt kasutuses on tähtis, et nende vaheline integratsioon oleks hästi läbi mõeldud ja töökindel.

3.1 Integratsioonide üldpilt

Commence'i ja OutSystemi pool realiseeritud integratsioonid võib kokku võtta *Joonis 8 - Commence ja OutSystems'i integratsioonid*, kus on näha järgmised ahelad:

1) Kollased integratsioonid, kus Commence klient-andmebaas saadab infot välja
    a. Outlook klient (kalender, kirjad)
    b. Commence server
    c. Välisveeb
    d. OutSystems
    e. Ärianalüüsi tarkvara
    f. Dokumendihaldussüsteem
    g. Microsoft Exchange
2) Punased integratsioonid, kus Commence'i server saadab infot Commence'i klientidele
    a. 300+ lõppkasutaja andmebaasi
    b. Statistika andmebaas
    c. Integratsiooni-andmebaas (OutSystems)
3) Rohelised integratsioonid, kus OutSystem saadab infot välja
    a. Office Online
    b. Dokumendihaldusüsteem
    c. Integratsiooni andmebaas (Commence)

d. Välisveeb

e. Ärianalüüsitarkvara

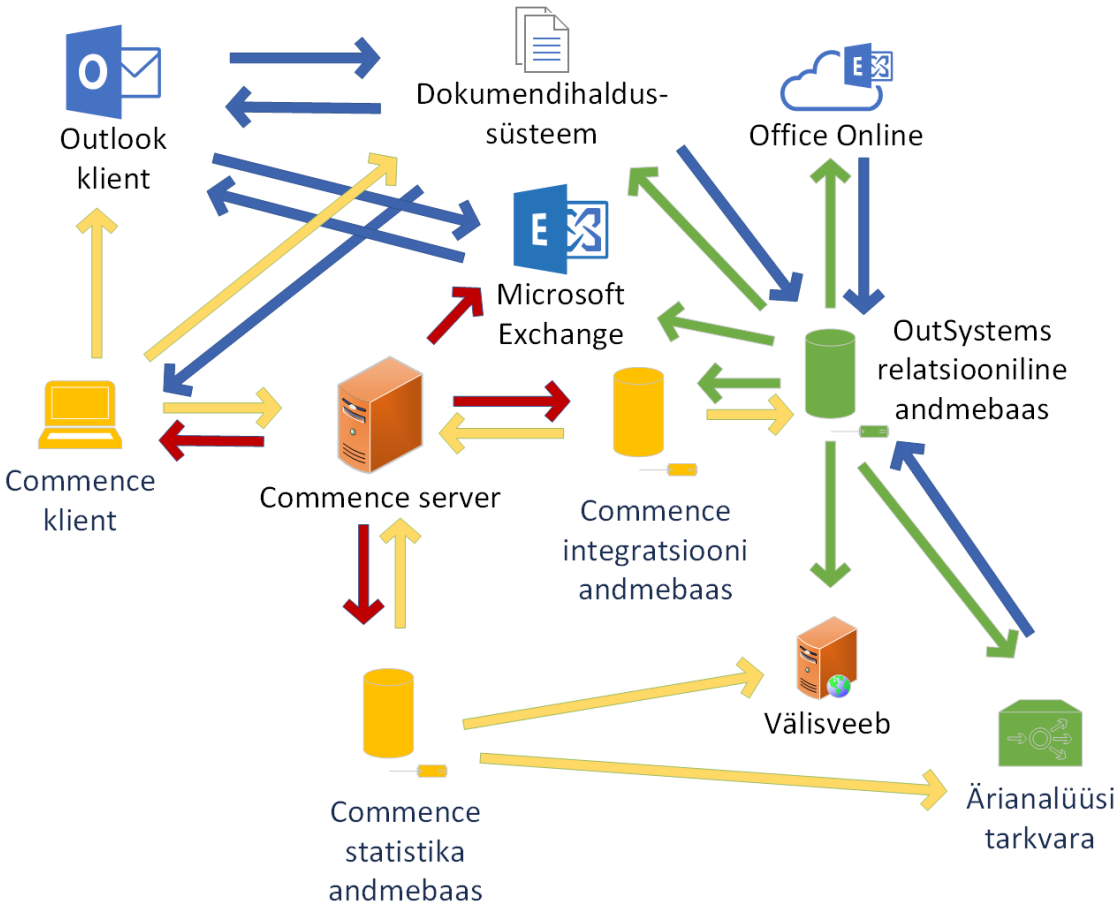
f. Microsoft Exchange

4) Sinised integratsioonid mis saadavad infot omavahel või Commence'ile / Outsystemile

a. Outlook klient < > Dokumendihaldussüteem

b. Outlook klient < > Microsoft Exchange

c. Dokumendihaldussüsteem < > Outsystems



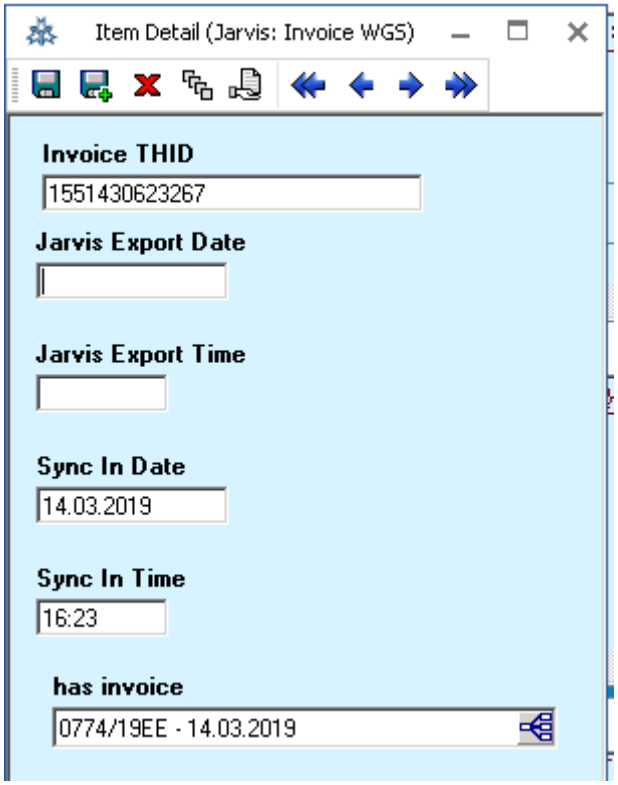
Joonis 8 - Commence ja OutSystems'i integratsioonid, autori joonis

3.2 Commence ja OutSystems integratsioon

Commence on paksu kliendi arhitektuuriga (*fat client arhitecture)* andmebaas. See tähendab praktikas, et neljas kontoris on üle 300 lokaalse andmebaasi. Kui muudatus teha

kliendi poolel siis see andmebaas saadab info serverisse üle FTP protokolli ja teised andmebaasid laadivad selle info alla.

Selleks, et Commence'is tehtud muudatused jõuaksid OutSystems'isse tegi autor iga kategooria kohta nö järjekorra kategooria kus on seos originaal-kategooriasse, ajatempel millal info integratsiooni andmebaasi jõudis ning ajatempel millal info sealt edasi OutSystem'isse saadeti. Seda iseloomustab Joonis 9 - Commence - OutSystems järjekorrakategooria kus on näha Commence'i sisene identifikaator ning nii Commence'i kui OutSystem'i ajatempel, samuti seos originaalkategoorias (arve). Vormil on kasutatud nime „Jarvis" mis on OutSystem'is loodud toote nimetus lõppkasutaja jaoks.



Joonis 9 - Commence - OutSystems järjekorrakategooria, autori joonis

Integratsiooni jaoks võttis autor kasutusele veel ühe Commence'i andmebaasi kust kõik olemasolevad- ning jooksvad lisatud/muudetud andmed VBS keeles OutSystem'i andmebaasi sisestas. Andmete muutmise Commence'is võib integratsiooni poole pealt kokku võtta järgmise töövooga (Lisa 3 - Commence'i poolt andemete muutmise töövoog):

1. Andmete algne lisamine:
2. Iga kategooria kohta
    a. Küsi Commence'ist kõik ühe kategooria kirjed ja lisa SQL'i

b. Küsi Commence’ist selle kategooria seosed ja lisa SQL’i

3. Andmed jõuavad integratsiooni andmebaasi (Commence)
    a. Lisa andmed sellele kategooriale vastavasse abitabelisse
    b. Lisa viide originaalile
    c. Kontrolli kas andmed on SQL andmebaasis
        a) Kui on, siis uuenda andmed ning üks-mitmesed ja mitu-mitmesed viited
        b) Kui ei ole, siis lisa andmed ning üks-mitmesed ja mitu-mitmesed viited
    d. Kontrolli, kas andmete lisamine / muutmine õnnestus
        a) Kui andmete lisamine / muutmine õnnestus (andmebaasi sünkroniseerimise ajatempel on uuenenud) täida Commence'i abitabelis sünkroniseerimise väli ajatempliga - antud kirje peidetakse järjekorrast
        b) Kui ei õnnestunud, teavita arendajat, ära täida Commence'i ajatemplit (jääb järjekorda).

Andmete muutmise OutSystemsis võib integratsiooni poole pealt kokku võtta järgmise töövooga (Lisa 4 – OutSystemi poolt andmete muutmise töövoog):

1. Täida andmete lisamine / muutmise lahter ajatempliga
2. Integratsiooni VBS kood kontrollib iga kategooria kohta kas on andmeid mis on muudetud pärast viimast sünkroniseerimise ajatemplit, kui on siis
3. Kontrolli kas andmed on Commence andmebaasis
    a) Kui on, siis uuenda andmed ning üks-mitmesed ja mitu-mitmesed viited
    b) Kui ei ole, siis lisa andmed ning üks-mitmesed ja mitu-mitmesed viited
4. Kontrolli, kas andmete lisamine / muutmine õnnestus
    a) Kui andmete lisamine / muutmine õnnestus (VBS kood ei tagastanud veateated) täida Outsystemsis vastava kirje sünkroniseerimise ajatempel

b) Kui ei õnnestunud, teavita arendajat, ära täida OutSystemis vastava kirje sünkroniseerimise ajatempel (jääb järjekorda).

Ülalkirjeldatud töövood tagavad, et esimesel korral on antud seisuga kõik andmed üle kantud Commence'is Jarvisesse ja edaspidi töödeldakse kas Commence'i või OutSystem'i poolt järjekorda lisatud kirjeid.

3.3 Integratsioon ärianalüüsi tarkvaraga (QlikView)

QlikView on ärianalüüsi tarkvara mis analüüsib erinevate allikate andmeid. Põhiline andmeallikas on Commence aga info tuleb ka raamatupidamistarkvarast, dokumendihaldussüsteemist, välisveebist ja Exceli failidest. Analüüsitud andmete põhjal koostab algoritm kas kord päevas või *ad-hock* aruandeid erinevates vormingutes (HTML, PDF, Excel). Ühtlasi saab QlikView klienti kasutada käsitsi andmeanalüüsiks uute raportite tegemiseks.

Selles etapis kaardistas autor olemasoleva integratsiooni (CSV formaadis tekstifailide ühesuunaline eksport) ning ehitas sarnase lahenduse (ajatatud töö) OutSystems'i poolel. Tabel 5 - Commence - QlikView integratsioon näitab, et Commence'is poolt toimub eksport 21 erinevas etapis ning juba kaardistamise hetkel selgus, et kõiki neid pole vaja. Sellisel kujul andmete eksport on ajamahukas ja kestab kokku ligi 5 tundi. Selleks, et tagada QlikViews õiged andmed tuleb luua sama väljundfail OutSystem'i pool ning seda siis Commence'i failiga võrrelda – kui see fail kattub on andmete terviklus tagatud.

Jrk.	Commence kategooria	Kas tuleb OutSystem'isse?
1.	Kaasus	Jah
2.	Klient	Jah
3.	Valuuta	Jah
4.	Töötaja	Jah
5.	Arve	Jah
6.	Tiim	Jah
7.	Tunnikaart	Jah
8.	Isik	Jah
9.	Tunnihind	Jah
10.	Kliendigrupp	Jah
11.	Valdkond	Jah
12.	Publikatsioonid	Ei
13.	Treeningsündmused	Jah

14.	Kliendilepingud	Jah
15.	Kontaktid	Ei
16.	Treeningud	Jah
17.	Maksed	Jah
18.	Kohtumised	Ei
19.	Kliendi toomine	Ei
20.	Kaasuse toomine	Ei
21.	Projektid	Jah

Tabel 5 - Commence - QlikView integratsioon, autori tabel

CSV formaadis eksport sai kunagi valitud kuna Commence'i puhul pole tegemist relatsioonilise andmebaasiga ning ärianalüüsi tarkvara ei saa muud moodi andmeid otse Commence'ist lugeda. Ärianalüüsi tarkvara saab tegelikult infot lugeda ka otse andmebaasist (eraldi vaadetega) või üle REST API ning nii on plaanis see ka Outsystem'i jaoks tulevikus realiseerida. See võimaldaks andemete lugemise teha kiiremaks ja töökindlamaks. Ühtlasi võimaldaks see lahendus saata edasi vaid muutusi, mitte kõiki andmeid.

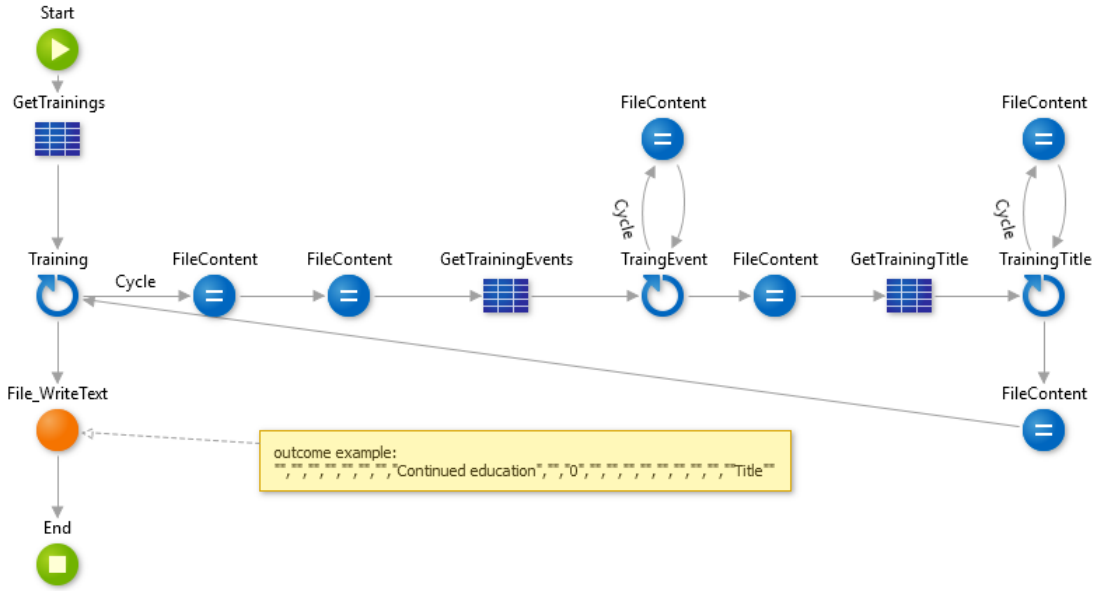
Esimeses etapis realiseeris autor OutSystemsi poolt CSV formaadis eksportimise nii, et QlikView poolt toimuks andmete import sama moodi edasi. Esimene moodul mis viidi täielikult OutSystem'isse üle ning pandi Commence'i pool lukku oli treeningud. Selleks, et saada Commence'i eksportfailiga sama tulemust tegi autor treeningute näitel alljärgneva ajatatud töö OutSystem poolel:

Küsi andmebaasist kõik treeningud. Iga treeningu kohta

a) Küsi kõik seotud treeningsündmused

b) Küsi kõik seotud treeningutüübid

Tulemuseks on iga treeningu kohta üks rida CSV failis, kus veerud eraldatakse semikooloniga ning kui mitme seoses korral eraldatakse seosed jutumärkide sees reavahetusega. OutSystemsi pool realiseeritud eksporti illustreerib *Joonis 10 - Outsystems - QlikView integratsioon (treeningud)*. Nii saab võrrelda, kas OutSystems'ist ja Commence'ist ekspordid andmepakett kattub. Seda eksporti saigi edaspidi kasutatud kuna muutusi enam Commence'isse tagasi ei saadetud ning antud moodul oli lõppkasutaja jaoks üldse külmutatud.

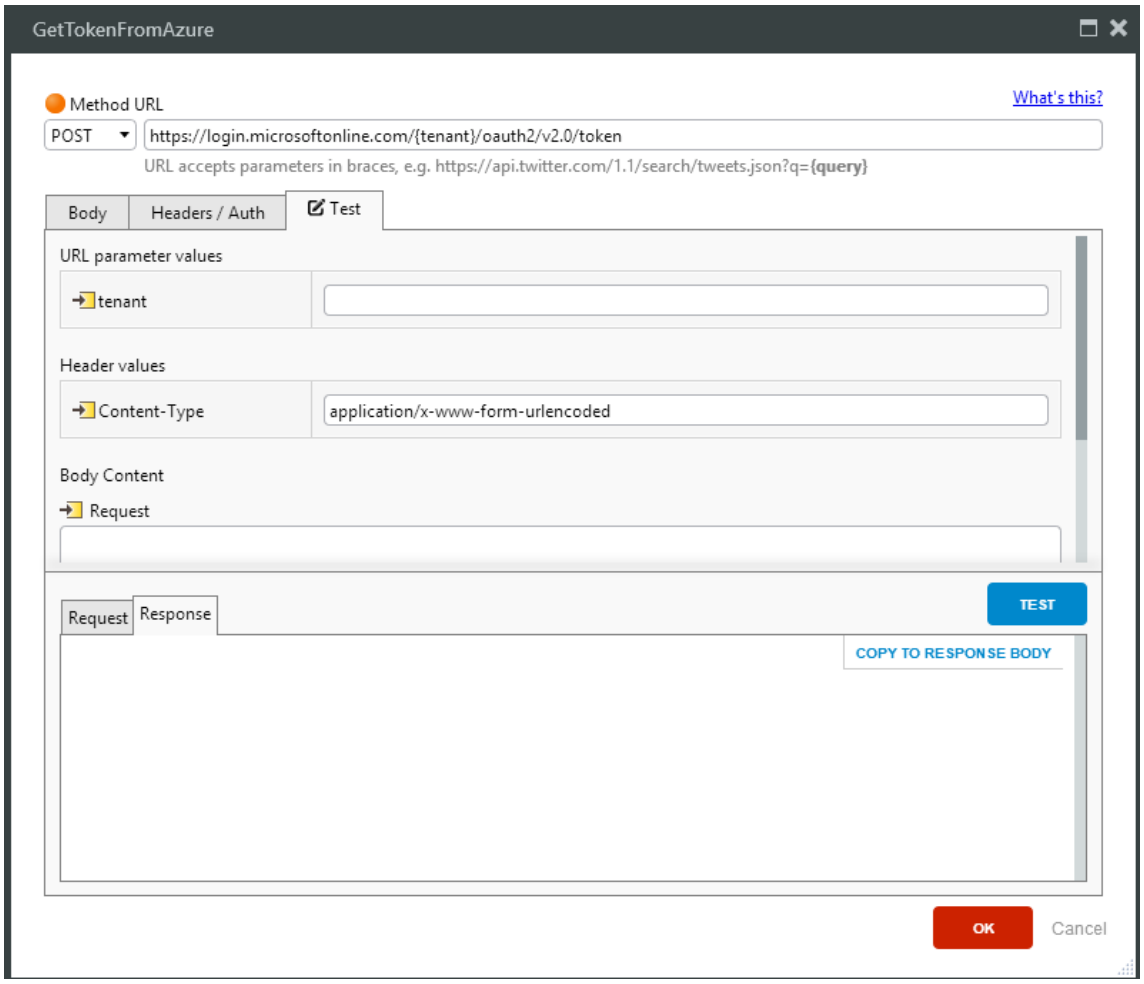


Joonis 10 - Outsystems - QlikView integratsioon (treeningud) , autori joonis

3.4 Integratsioon Microsoft Exchange'iga.

Selles etapis realiseeris autor OutSystems integratsiooni Microsoft Exchange'iga ja Office 365'ga. Paralleelselt Commence'i migreerimisega Outsystem'i platvormile migreeris autor koos kolleegidega e-posti serveri Microsoft Exchange ettevõtte serveripargist Exchange Online'i peale. Seega võimaldas autoril implementeerida juba uue põlvkonna andmevahetus (Graph API), mis asendas vana VBS koodi.

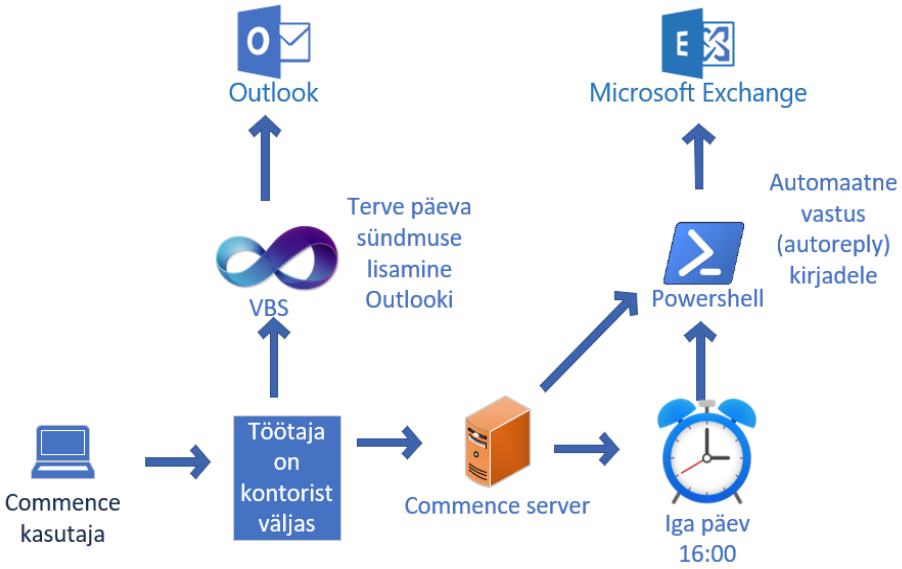
Selleks, et OutSystems saaks suhelda Office 365'iga on vaja kasutaja kõigepealt tuvastada. See kujutab endast POST meetodiga login.microsoftonline.com poole pöördumist *Joonis 11 – Kliendi autoriseerimine Microsoft Graph API kaudu*



Joonis 11 – Kliendi autoriseerimine Microsoft Graph API kaudu, autori joonis

Töövoog on selline:

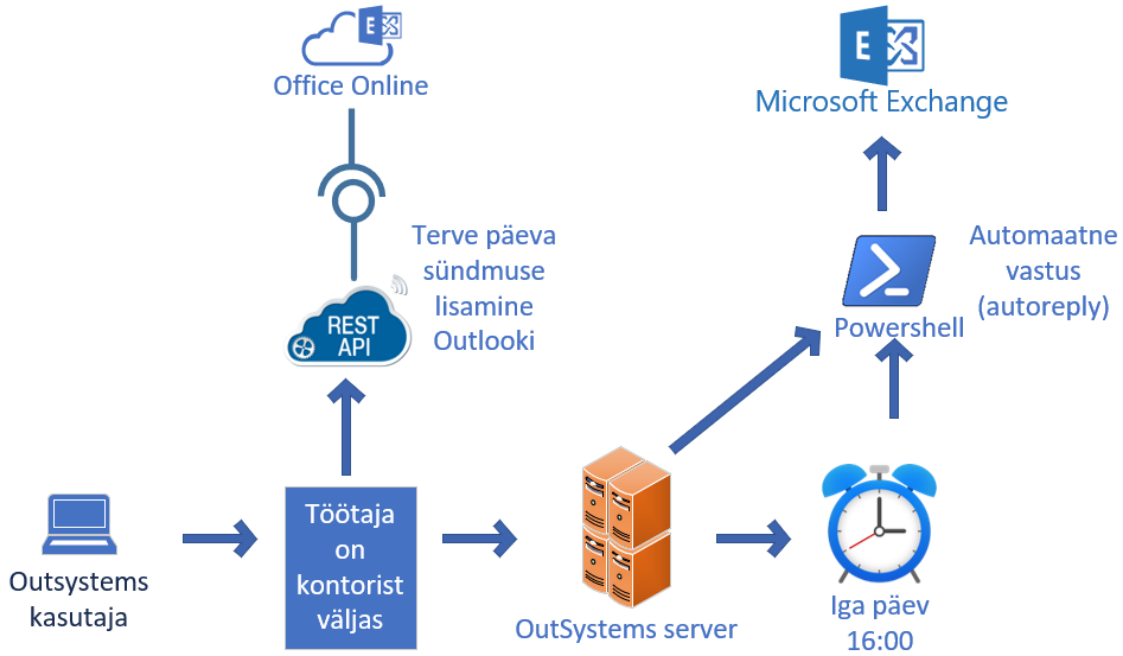
1. Personaliosakond saab taotluse, et töötaja soovib kontorist eemal olla (nt. puhkus).
2. Taotlus lisatakse süsteemi
3. Commence'i korral (Joonis 2 - Commence - Exchange integratsioon)



Joonis 2 - Commence - Exchange integratsioon, autori joonis

a. VBS kood lisab kohe kalendrisündmuse Outlooki. Kui teade lisatakse korraga mitmele inimesele (riiklikud pühad) siis korratakse protsessi iga inimese kohta.
b. Kirje sünkroniseeritakse serverisse

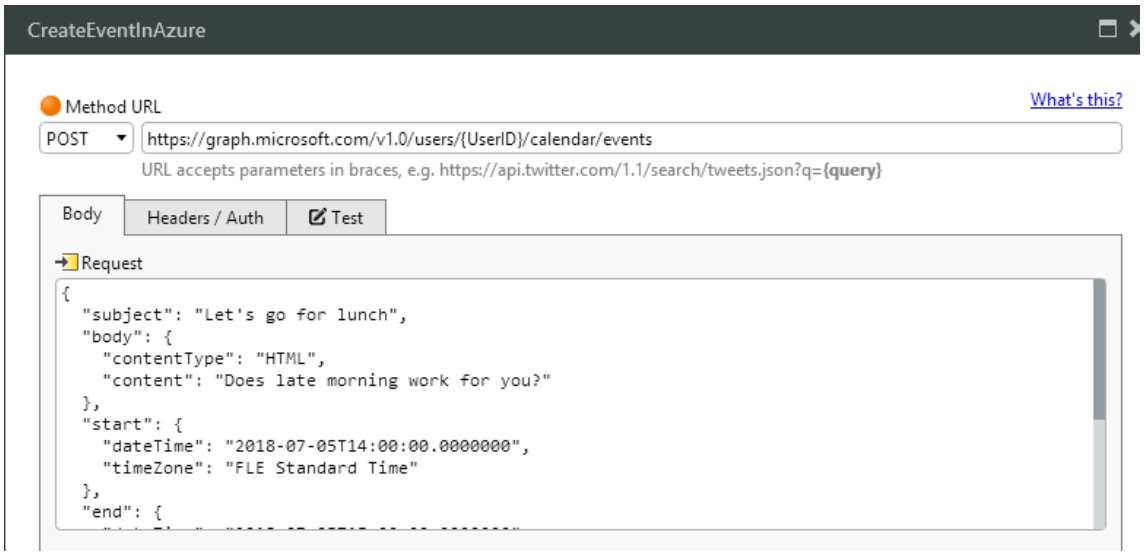
4. OutSystemsi korral (Joonis 12 – Outsystems – Exchange integratsioon)



Joonis 12 – Outsystems – Exchange integratsioon, autori joonis

a. Graph API abil lisatakse kohe kalendrisündmuse postkastile (Joonis 13 – Kalendrikirje loomine Microsoft Graph API kaudu, autori joonis). Kui

teade lisatakse korraga mitmele inimesele (riiklikud pühad) siis korratakse protsessi iga inimese kohta. Selleks otsitakse kõigepealt e-maili järgi välja kasutaja ID ning selle alusel luuakse kalendrikirje.



Joonis 13 – Kalendrikirje loomine Microsoft Graph API kaudu, autori joonis

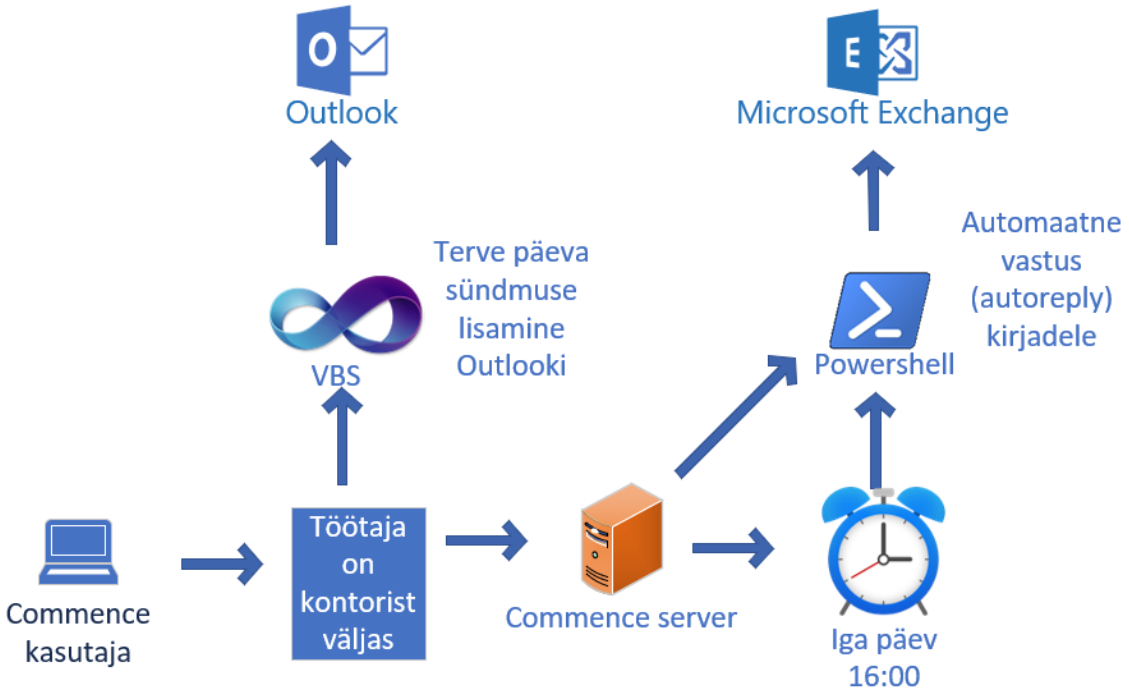
5. Mõlemal süsteemi korral - kui taotluse alguskuupäev on käesolev päev siis ekspordikase vastav info (CSV) kohe Exchange'i (enne lokaalne Exhange, nüüd Exchange Online) jaoks tekstifaili ning antakse süsteemile korraldus PowerShell'i abil kontorist väljasoleku teade aktiveerida.
6. Mõlemal süsteemi korral - iga päev kell 16 kontrollitakse kes järgmisel tööpäeval kontorist eemal on. Need kirjed eksporditakse ning automaatika kontrollib, kas kontorist eemaloleku teade on juba rakendatud (oli ka eile nimekirjas). Kui mitte, siis aktiveeritakse kontorist eemaloleku teade. Ühtlasi kontrollitakse neid, kellel on juba teade aktiveeritud – kas nad on ka uues nimekirjas. Kui ei ole, võetakse nende teade maha. Kogus see PowerShell kood on autori looming.

3.5 Integratsioon välisveebiga (www.sorainen.com)

Selles etapis arutas autor veebilehe arendajatega kuidas realiseerida detailselt töötajate info saatmist välisveebi ning seadistas OutSystem'i ja välisveebi vahel reaalajas REST liidestuse.

Büroo veebileht sai seni enamuse infost Commence'i töötaja profiili kaudu. Kord päevas eksporditi see Commence'ist CSV formaadis välja ning server importis ajatatud töö

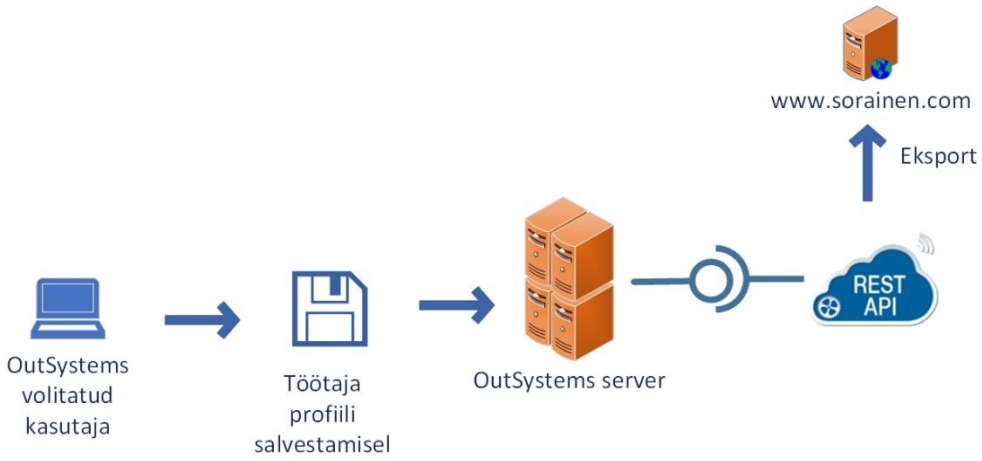
raames (*cron job)* andmed MySQL andmebaasi. Importimisega tegeles PHP kood mis ühtlasi valideeris ka CSV faili sisu on korrektne ning kõik vajalikud veerud on täidetud. Kui kohustuslik väli oli puudu jättis import oma töö katki ning saatis autorile teate koos kirjeldusega – millised andmed on vigased.



Joonis 14 – Commence – välisveeb integratsioon, autori joonis

Sellist liidestust OutSystems'i jaoks enam vaja pole, sest paralleelselt juurutati uus välisveebi platvormi, mis võimaldab andmed tarbida REST API kaudu. OutSystem pakub neid andmeid teenusena. Kui vana välise veebi server asub ettevõtte serveripargis siis uus platvorm on majutatud välise teenusepakkuja juures. See omakorda tähendab, et andmete vahetus tuleb tagada krüpteeritud kujul turvatunneli kaudu. Tõsi, veebis nähtavad andmed on avalikud, aga me ei taha, et keegi kolmas osapool sinna vahele saaks ning meie eest andmeid veebiserverile esitleks.

Realiseeritud integratsiooni kujutab *Joonis 15 - OutSystems integratsioon uue veebilehega* ning selle kohaselt toimub kohe peale töötaja profiili muutmist muudatused saatmine REST API kaudu välisveebi. Plaanis on luua ka eraldi järjekord, et kontrollida andmete saatmise edukat lõpetamist – kui andmeid ei õnnestu saata (server ei tagasta kinnitust) siis muudatus järjekorrast ära ei kao. Hetkel järjekorda pole ning kui viga tagastatakse siis saadetakse automaatne teade IT kasutajatoele.

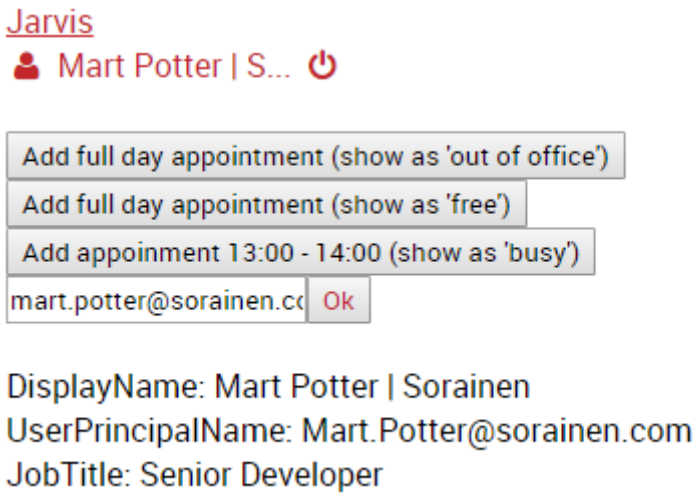


Joonis 15 - OutSystems integratsioon uue veebilehega, autori joonis

3.6 Integratsioonide testimine

Kui integratsioonid on realiseeritud on hädavajalik teha ka integratsiooni testid (Ingl. *Integration tests*). Kombineerides integratsioonitesti visuaalse testiga saab kontrollida, kas andmete terviklus on tagatud [21].

Välisveebi integratsiooni saab kontrollida vaadates kas OutSystem'is töötaja profiili all olnud andmed on samad mis REST API'ga välisveebi saadetud profiilil kuvatakse. Microsoft Exchange'iga integratsiooni saab kontrollida proovides lisada kalendrikirjet iseendale ning siis seda oma Outlookist otsides. Selle testimiseks tegi autori lihtsa leheküljе (Joonis 16 - Graph API testimise leht, autori joonis), mis otsib Graph API kaudu e-mail järgi inimese, näitab tema profiili ning võimaldab lisada erinevaid kalendrisündmusi.



Joonis 16 - Graph API testimise leht, autori joonis

3.7 Integratsioonide kokkuvõte

Integratsioonide osas kaardistas autor olemasolevad Commence'i integratsioonis ja realiseerisid osad neid ka OutSystem'i platvormil. Kui teatud integratsioonid, nt ärianalüüsi tarkvaraga, jätkuvad endiselt CSV failide tasemel siis päris mitmed integratsioonid sai autor OutSystem'i pool realiseerida mikroteenuste arhitektuurist lähtuvalt REST API't kasutades. Autor leidis, et REST API võimaldab teha integratsiooni andmevahetuse intervalli oluliselt tihedamaks (kord päevas asemel siis, kui olem lisatakse või muudetakse). Samas leidis autor, et integratsioonid töökindluse tõstmiseks peaks realiseerima tulevikus muudatuste järjekorra mida töödeldes saaks tõsta veakindlust. Ehk siis kui API ühendus ei ole edukas (nt teise osapoole teenuses on tõrge) siis ei jää kirjed edastamata vaid proovitakse hiljem uuesti. Autoril on plaanis selline järjekorrasüsteemi realiseerida kui migratsiooniprojekt on läbi.

4 Andmete turvalisuse tagamine tagamine

Kolmik konfidentsiaalsus – käideldavus - terviklus on moodustavad nö CIA kolmnurga (*CIA triad*). Selleks, et mitte segamini ajada seda mõistet Luure Keskagentuuri lühendiga siis kutsukse antud kolmikut ka teises järjekorras *AIC triad* (Joonis 17 - CIA (AIC) triad).. Antud kolm elementi on turvalisuse kõige kriitilisemad komponendid. [28]

Joonis 17 - CIA (AIC) triad [28]

4.1 Käideldavus

Käideldavus on üks kolmest teabe turvalisuse tuumelemendist (teised kaks on konfidentsiaalsus ja terviklus); iseloomustab teabe, IT-süsteemide, inimeste ja protsesside teovõimet ja kättesaadavust sel ajal, mil organisatsioon neid vajab; sõltuvalt kontekstist on tähendus kvalitatiivne ("on/ei ole käideldav") või kvantitatiivne ("mil määral on käideldav") [24].

4.1.1 Käideldavus Commence'is

Commence'is võis käideldavus hea omadusena esile tuua seda, et paksu kliendi arhitektuurist tingituna on kasutajal kõik andmed teatud seisuga olemas internetiühendusest olenemata.

Samas kasutaja tehtud muudatused jõuavad teiste kasutajateni parimal juhul poole tunniga aga erinevate asjaolude kokkulangemisel võib infovahetus aega võtta päevi või nädalaid. Seega aga tähendab, et kui üks osapool on arveldaja ning teine osapool peab sisse kandma oma töötunnid võib tulemuseks olla poolik arve kuna kõik töötunnid pole ühest Commence'ist teise õigeks ajaks jõudnud. Võimalikud stsenaariumid on, et pärast oma töö kirjapanekut pannakse Commence'i rakendus kinni, tehaksegi tööd ilma internetiühenduseta või ilma VPN'i ühendus loomata.

Selleks, et Commence'i käideldavuse riski maandada on autori loonud monitooringusüsteemi mis jälgib Commence'i poolset serveri seisu ning teeb siis tulemusest visuaalse tabeli kus näidakse ära kasutajad kelle Commence'i andmebaas pole juba mõnda aega serveriga suhelnud. Sellest tingituna võib kohati olla ka suur käideldavuskadu – nt kui tuleb kontrollida, kas antud kliendi tööd võib vastu võtta või mitte ning kõik andmed ei ole Commence'isse jõudnud võib vale otsuse vastu võtta.

Joonis 18 - Commence monitooringu näide illusteerib, et üks andmebaas pole serveriga ühendust saanud alates veebruaris ning osadel kasutajatel on väga palju infot puudu. Ühe sünkroniseerimise sessiooniga saadetakse umbes 15'000 ühikut infot, seega andmebaas millel on 1'000'000 ühikut puudu on lootusetult maha jäänud ja tuleb uuesti teha.

before last w-day

#	user	date	time
1.	vaiva.v	15.02	18:39
2.	jane.e	14.03	15:51
3.	kadi.l	14.03	16:46

most behind

#	behind	user	date	active.log
1.	1592121	katrina.b	14.05 17:34	14.05.2019 18:00
2.	1201681	vaiva.v	15.02 18:39	15.02.2019 19:00
3.	938765	irina.s	11.04 12:04	14.05.2019 18:00

Joonis 18 - Commence monitooringu näide, autori joonis

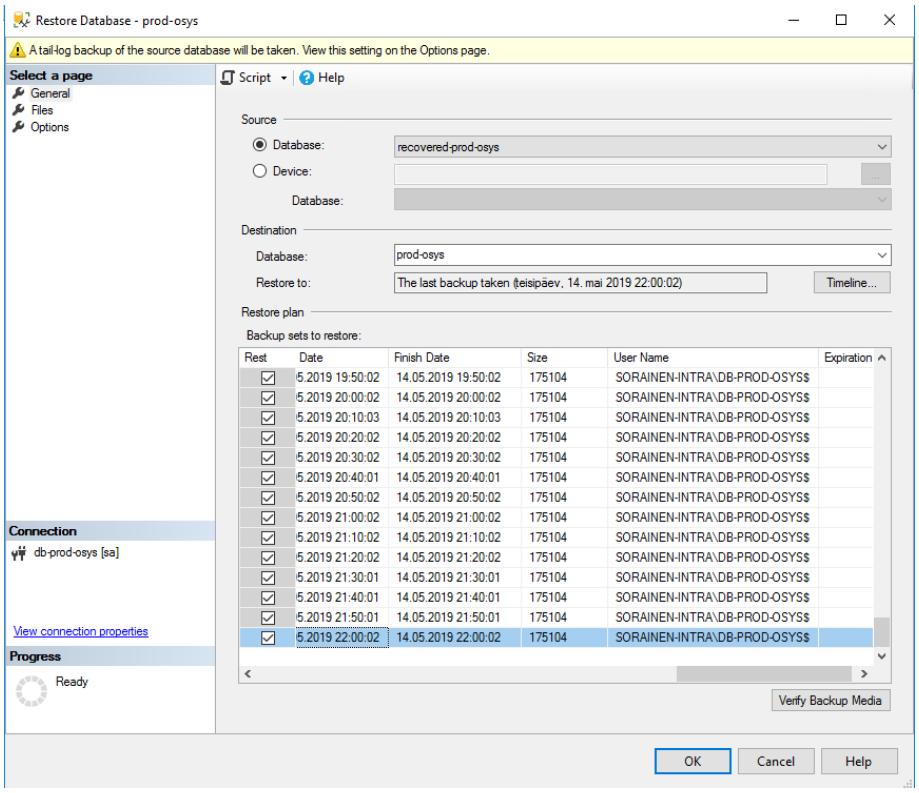
Üks osa käideldavusest on ka andmete taasteplaan. Commence'i puhul tehakse üks kord päevas ühes Commence'i andmebaasist koopia eraldi faili mida on vajadusel võimalik taastada. Commence'ist on võimalik ka andmeid välja eksportida sellises formaadis, et sama seadistust kasutades on võimalik andmed teise Commence'i andmebaasi importida. Samas on see viide siiski liiga suur (kuni 24h) ning sedagi vaid juhul kui kasutaja muutus on koopia tegemise ajaks serverisse jõudnud.

4.1.2 Käideldavus OutSystem'is

OutSystems on reaalaja süsteem ehk kui üks osapool on oma sisendi andud OutSystem'is siis teine näeb seda OutSystem'is kohe. Seni kuni kogu funktsionaalsus pole aga Commence'is üle toodud on endiselt viide nii arveldamises kui ka näiteks andmeanalüüsis. Enamus andmeid eksporditakse analüüsi tarbeks endiselt Commence'ist kuna *master-data* on just seal. OutSystem'i pool annab käideldavust juurde ka asjaolu, et osadele andmetele (nt oma tööaja kirjapanek) saab ligi ka nutiseadme abil (VPN turvatunnelit kasutades). OutSystem'ist sisestatud / muudetud andmed jõuavad kasutajast olenemata arveldajani, st kui seal on viide, siis see pole kasutajast tingitud. Lisaks on ühe kasutaja muudatuste kohe teistele sama platvormi kasutajatele kättesaadavad – st nii kui isik A loob uue projekti saab isik B hakata sinna oma aega registreerima – Commece'i puhul oli see mina poole tunnise viitega, tõenäoliselt aga rohkem.

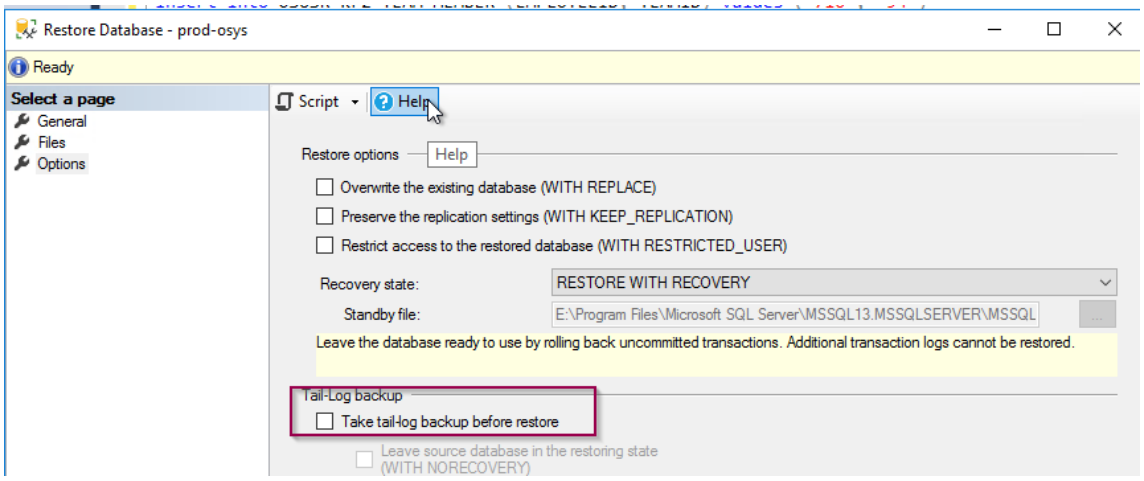
Taasteplaanina on OutSystem'i puhul kasutatud Microsoft SQL'i võimekust teha kogu andmebaasist varukoopia kord päevaks. Sellele lisaks tehakse iga kümne mitu tagant varukoopia vaid muudatustest, ehk kui leiti mingi viga siis on võimalik teatud tabelisse taastada andmed mis olid seal 10 minutit tagasi. Selleks, et võtta andmed varukoopiast tuleb kõigepealt luua uus tühi andmebaas ning siis taastada sinna *LIVE* andmebaasi varukoopia soovitud seis.

Selleks, et võtta andmed varukoopiast tuleb kõigepealt luua uus tühi andmebaas ning siis taastada sinna *LIVE* andmebaasi varukoopia soovitud seis. Leitud varukoopiate valikus tuleb leida endale sobiv nii nagu on näha Joonis 19 -Microsoft SQL varukoopia valik, autori joonis



Joonis 19 -Microsoft SQL varukoopia valik, autori joonis

Enne kui taastama hakata peaks kindlasti „Options“ alt valima, et *Tail log backup*-i taastada pole vaja nii nagu on näidatud Joonis 20 - Microsoft SQL varukoopia logi välistamine, autori joonis



Joonis 20 - Microsoft SQL varukoopia logi välistamine, autori joonis

Kui varukoopia on uude andmebaasi taastatud saab juba SQL käskudega andmed ühest andmebaasist teise tõsta. Näiteks kui sooviks taastada andmebaasis [prod-osys] kustutatud arve CMCID alusel siis saaks seda teha järgmise SQL koodiga (autori kood)

```
INSERT INTO [prod-osys].dbo.OSUSR_EW4_INVOICE (InvoiceNo, PaidOn, EURunpaid,
ToEntity, CREATEDAT,CMCID) SELECT InvoiceNo, PaidOn, EURunpaid, ToEntity,
CREATEDAT FROM [prod-osys-recovered] WHERE CMCID '13:8004E501:FFFEC937');
```

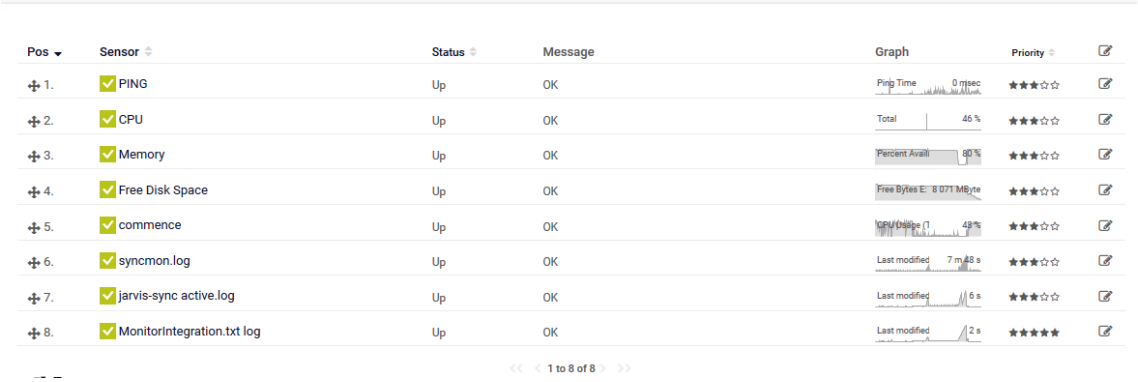
4.1.3 Käideldavus migratsiooni käigus

Tingituna asjaolust, et üsna pikalt on mõlemad süsteemid paralleelselt kasutuses siis peab autor tagama, et andmed liiguksid piisavalt kiiresti ja tõrgeteta Commence'ist OutSystem'isse ja vastupidi. Näiteks seni, kui klienti ja projekte sai luua vaid Commence'is tuli oodata, et loodud klient / projekt jõuaks OutSystem'isse, et siis projektile tunnikaarte (tehtud tööaega) registreerima saaks hakata. Teisalt aga seni, kuni arvete tegemine on endiselt Commence'is tuleb tagada, et needsamad tehtud tunnikaardid jõuaksid õigeks ajaks OutSystems'ist tagasi Commence'isse, et nende põhjal siis arveid teha.

Ühelt poolt on tähtis, et andmeid liigutakse ühest süsteemist teise õiges järjekorras (kõigepealt klient, siis projekt), teisalt see, et kui tekib ka mingi viga siis saab keegi sellest kohe teada. Vigu võib olla erinevaid, alates andmete formaadi mittesobivuses (oodatakse sisendiks numbrit aga tuleb tekst) või kogu protsessi teenindava virtuaalmasina jõudlusest (mälu ülekasutus).

Pika seisakuaja risk (ingl. *Extended Downtime Risk*) ilmneb kui andmete migreerimise protsess võtab ettenähtust rohkem aega [22]. Selleks, et antud probleemidele piisavalt kiiresti reageerida on autor seadistanud erinevaid mõõdikuid ning liidestanud need ettevõtte PRTG monitooringusüsteemiga.

Mõõdikud võib laias laastus jagada kaheks – virtuaalmasina üldine tervis ja teatud failide muutmise aja monitooring. Kõikide mõõdikute ülevaadet saab näha Joonis 21 - OutSystems integratsiooni mõõdikute ülevaade, autori joonis.



Joonis 21 - OutSystems integratsiooni mõõdikute ülevaade, autori joonis

Virtuaalmasinas kohta kogub autor järgmisi andmeid:

1. *PING* – kas virtuaalmasin on üle võrgu pidevalt kättesaadav
2. *CPU* – protsessori koormus
3. *Memory* – mälukasutus
4. *Free Disk Space* – vaba kettaruumi olemasolu

Integratsiooniga seoses kogu autor PRTG jaoks järgmisi andmeid

5. *Commence* – kas Commence'i protsess käib
6. *Synclog.mon* – kas Commence klient on Commence'i serveriga andmeid vahetanud
7. *Jarvis-sync active.log* – kas Commence klient on erinevaid operatsioone teinud (või kokku jooksnud)
8. *Monitormigration.txt log* – kas Commence ja Jarvis on infot vahetanud või on see mingi vea tõttu seisma jäänud.

Igal mõõdikul on teatud veapiirid paika pandud – mis on lubatud ja millal tuleb autorit alarmeerida. Näiteks kui kettaruumi on vähem kui 2GB või kui Monitormigration.txt failis pole pool tundi mingeid muutusi olnud. Edasi on autor defineerinud, et olenevalt probleemi tõsidusest annab PRTG teada kas e-maili või sms'i teel, et kõnealune sensor annab häiret. Riski maandamiseks antakse häiret mitmele inimesele. Ühtlasi annab PRTG ka teada kui sensor on taas lubatud piirides (seis stabiliseerub).

Lisa 6– PRTG sensori MonitorIntegration.txt log graafik näitab ilmekalt ära, et viimase 24 tunni jooksul on sensor alarmi andud ning töö seisma jäänud. Kuna antud juhul tekkis viga laupäeva hommikul siis ilma sellise monitooringute pole keegi viga enne järgmist tööpäeva märganud ning kogu integratsioon oleks seisnud.

4.2 Terviklus

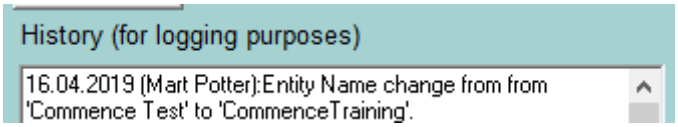
Lähtudes teadmisest, et pärandsüsteemi (Commence) ning uue süsteemi (OutSystems) andmebaasi arhitektuur on erinev tekib vajadus viia andmed ühelt kujult teisele nii, et andmete terviklus säiliks. Selleks on vaja välja töötata õige strateegia.

AKI [15] selgitab ,et terviklus pole infoturbes oluliselt erineva tähenduse tõttu sama mis terviklikkus. Tervikluse olemus:

1. ISO/IEC 27000: õigsus ja täielikkus:
    - infovara lubamatute muudatuste puudumine
    - hõlmab ka autentsust ja salgamatust
    - üks teabe turvalisuse kolmest põhikomponendist levinuimas turvamudelis
2. ISO/TR 13569: varade õigsuse ja täielikkuse kaitstus

4.2.1 Terviklus Commence'is

Antud projekti kontekstis ei saa me paraku eeldada, et Commence'is oleks andmed seotud selle looja ja/või loomise ajaga kuna andmebaas sellist infot eraldi ei talletanud. Teatud juhtudel kirjutati selline logi eraldi teksti kasti aga selle peale lootma jääda ei saa (mõnikord on see logi segamini muu informatiivse tekstiga ning teksti on võimalik muuta, s.h. logi eemaldada). Sellise logmise näide on näha Joonis 22 - Commence logimise näide.



Joonis 22 - Commence logimise näide, autori joonis

Loomulikult on suur käideldavuskadu siis, kui kasutaja pole oma Commence'i klienti käivitanud nii, et samal ajal on loodud ühendus ka ettevõtte võrguga.

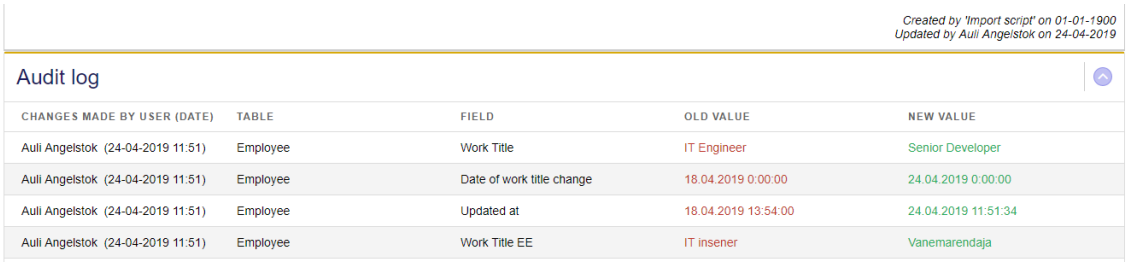
Viimaks, isegi kui kasutaja on püüdlikult oma Commence'it töös hoidnud ei saa alati ka kindel olla, et ta näeb sama infot nagu teised. See tähendab, et andmebaas on katki ning ei saa serverist enam uut infot kätte või ei saada uut infot välja – kui selline käideldavuskadu tekib, tuleb see andmebaas ära kustutada ning uus genereerida. Kui

kasutaja on teinud teatud uuendusi mida teised ei näe on suhteliselt lihtsate vahenditega võimalik see info välja eksportida ning töötavasse andmebaasi importida.

4.2.2 Terviklus OutSystem'is

Kuna OutSystemsi puhul on tegemist reaalaja infosüsteemiga siis on kõik kasutajad käideldavuse osas võrdsed ehk pole tähtis kui tihti kasutaja OutSystem'i rakendust külastab – alati on tema käsutuses kõige värskem info.

Looja ning loomisaeg registreeritud OutSystems pool. Samamoodi pannakse kirja kes ja millal mingi kirje muutusi. Peale loomise ja muutmise ajatempli on arendatud ka muudatuste auditeerimise moodul mis näitab kõikide kasutajate iga kirje või seose muutmise ajalugu vastavalt volitusele nagu on näha Joonis 23 - OutSystems audit logi näide.



Joonis 23 - OutSystems audit logi näide, autori joonis

4.2.3 Terviklus migratsiooni käigus

Andmete migratsiooni käigus võib ette tulete andmekao risk (Ingl. *Data Loss Risk*). Kui andmeid migreeritakse uude süsteemi või osa andmetest jääda üle kandmata. Seda riski saab vältida kui viia andmete migratsiooni testimine [22].

Selleks, et võrrelda Commence'i ja OutSystem'i iga kategooria andmeid kirjutas autor automaattesti prototüübi taandades mõlema andmebaasi tabeli kirjed sarnasele kujule.

Ajatatud töö kontrollib kõigepealt kas järjekorda on lisatud midagi (kirje millel puudub OutSystem'i ajatempel) ning lisab / muudab / kustutab andmed OutSystem'i andmebaasist ning uuendab kirje sünkroniseerimise ajatemplit Outsystems'i andmebaasis.

Outsystemsi ajatempli uuendamine ei toimu aga igal juhul vaid ainult siis, kui tehakse kindlaks, et SQL lisamise / muutmise ajatempel on olemas. Selleks kasutas autor SQL kuupäevaveergi *CREATEDAT* ja *UPDATEDAT* mis on oma olemusel DATETIME tüüpi. Uue kirje lisamisel omistati kirje *CREATEDAT = GETDATE(),* olemasoleva kirje muutmisel omistati *UPDATEDAT* = GETDATE().

Tallinna Tehnikaülikooli näitel näeks uue arve lisamine välja nii:

```
INSERT INTO OSUSR_EW4_INVOICE (InvoiceNo, PaidOn, EURunpaid, ToEntity,
CREATEDAT,CMCID) VALUES (N'0774/19EE', '2019-03-14', '1000', N'Tallinna
Tehnikaülikool', GETDATE(), '13:8004E501:FFFEC937');
```

CMCID on Commence unikaalne ID (antud juhul '13:8004E501:FFFEC937'). Selle järgi saab OutSystems Commence'isse kirjed tagasi saates teada, kas Commence'i kirjed peab uuendama (CMCID on olemas) või lisama. Lisamisel küsitakse loodud kirje CMCID ning uuendatakse see kohe ka kirjet OutSystemsis.

Kui nüüd Commence'is märgiti Arve 0774/19EE makstuks ja see arve on juba OutSystemsis olemas tuleb arve uuesti järjekorda ning OutSystem'i poole saadetakse *UPDATE* käsk. Selleks, et aga teada mida uuendada, otsitakse üles arve mille CMCID on meil teada.

```
SELECT ID FROM OSUSR_EW4_INVOICE WHERE CMCID='13:8004E501:FFFEC937';
```

Nüüd on teada, millist ID peab uuendama ning peale uue info arve kohta lisatakse ka uuendamise ajatempel.

```
UPDATE OSUSR_EW4_INVOICE SET PaidOn = '2019-03-19', EURUNPAID = '0',
UPDATEDAT =  GETDATE() WHERE ID='289542';
```

Kui SQL käsud on edukalt käivitatud siis saab peaksid vastava kirje *CREATEDAT* või *UPDATEDAT* olema muutunud.

Edasi kontrollib autor juba, kas ajatempli erinevus on piisavalt väike võrreldes hetkeajaga:

```
SELECT top 1 DATEDIFF (millisecond, CREATEDAT, GETDATE()) FROM
OSUSR_EW4_INVOICE with (nolock) WHERE cmcid='13:8004E501:FFFEC937';
SELECT top 1 DATEDIFF (millisecond, UPDATEDAT, GETDATE()) FROM
OSUSR_EW4_INVOICE with (nolock) WHERE cmcid='13:8004E501:FFFEC937';
```

Tulemuseks on ajavahe millisekundites mida saab siis võrrelda soovitud numbriga. Kui vahe on piisavalt väike siis täidetakse Commence'i ära OutSystems ajatempel hetke kuupäeva ja kellajaga.

Microsoft SQL dokumentatsiooni järgi [3] peaks saama iga *insert* / *update* käsku järgi kontrollida kas rea lisamine / muutmine õnnestus või mitte. Selleks peaks kohe pärast *INSERT* / *SELECT* käsku saatma SQL'i järgmise rea:

```
@@ROWCOUNT
```

Praktika näitab aga, et isegi kui rea lisamine või uuendamine oli edukas siis ei tagasta antud päring alati 1 vaid tihti on tulemuseks siis 0 (võimalik, et liialt väike viide), seega selle peale lootma jääda ei saa.

Teine võimalus on SQL vead kinni püüda VBS koodi sees. SQL'iga suhtlemiseks kirjutati ExecSQL funktsiooni kasutades ära VBS veahaldust (Lisa 7 – ExecuteSQL VBS funktsioon), Kui kasutada On Error Resume Next üksi siis ei jää vähemalt kood seisma kui on viga, samas ei saa keegi ka teada, et viga oli. Kui aga kontrollida vea koodi (Err.Number) saab teada, kas VBS suutis edukalt SQL käsu edastada. Antud juhul saab siis teada, kas kirjutatud SQL koodis olid süntaksi vead (nt puuduv sulg) aga mitte seda, kas tegelikult andmed lisati või uuendati. Nt kui andmetüüp ei klapi (*interger* vs *double*, *date* vs *text*) siis saab küsida Err.Description'it

Kui VBS'is jätta üldse On Error Resume Next ära ning koodis on viga, siis jääbki kogu kood seisma. Kui aga keegi seda ei märka siis võib ühe vea tõttu korrektsed andmed ka töötlemata jätta ning järjekord kasvab liiga suureks.

Orkestreerimisriski (Ingl. *Orchestration Risk*) vältimiseks on mõlemas suunas andmete sünkroniseerimisel tähtis kategooriate järjekord arvestades sünkroniseerimisel saadetud seoste olemasolu [22]. Näiteks tuleb kõigepealt saata kliendi ning siis projekti info kuna projekti on alati seotud täpselt ühe kliendiga.

4.2.4 Tervikluse valideerimine

Selleks, et valideerida, kas andmed on edukalt ületoodud saab erinevaid teste. *iCEDQ inside* on andmete migratsiooni testimise tehnikaid uurides leidnud, et alustama peaks täielikkuse testimiseks (*Completeness Tests*) kontrollimaks kas kõik andmed on üle toodud [21]. Seega enne kui hakata võrdlema andmete sisu teeb autor kindaks, et andmeid on sama palju. Commence'i puhul tähendab see töötajate tabeli näitel järgmist käsku:

```
Dim category : Set category=cmcdb.GetCursor(0,“Employees of SLO“,256)

category.RowCount()
```

Muutuja *category* on siinkohal Commence'i kategooria ning *RowCount* tagastab kõikide kirjete arvu juhul kui ei ole ühtegi filtrit lisatud. Microsoft SQL puhul loeb read kokku käsk *count()* [5]

Vastavalt juhendile loen üle kõik kirjed töötajate tabelis:

```
select count(*) from OSUSR_KF2_EMPLOYEE
```

Esimesel katsel need numbrid ei klappinud seega kirjutas autor automaattesti prototüübi, mis käis läbi kõik töötajad Commence'is ning kontrollis ka sellise Commence'i ID'le (CMCID) kirje on OutSystemis olemas ning nimi klapib. Seejärel käis kood läbi kõik kirjed OutSystem'is ning otsis kas CMCID'le vastav kirje on Commence'is olemas ning nimi klapib. Tulemuseks oli üks vigane kirje Commence'i poole peal mis ei pidanudki OutSystemis olema. Pärast selle kirje eemaldamist numbrid klappisid.

iCEDQ inside leiab, et kui andmete hulk on sama, tasuks tegeleda ka visuaalse testimisega (*Appearance Tests*) vaadates käsitsi üle kirjed nii vanas kui uues süsteemis tegemaks kindlaks, kas andmed näevad ka välja sellised nagu nad välja nägema peaks [21].

Edasi tuleb tagada, et nii mõlema süsteemi regionaalsed seaded on samad. Näiteks kui SQL serveri seades on kasutatud UK inglise keelt (A..Z) siis peab tööjaamas kus eksporditakse pärandsüsteemi infot olema kasutatud sama keelt. Kui kasutada nt eestikeelset seadistust siis on tähestiku järjekord erinev ning sorteeritud andmed pole võrreldav.

Seejärel tuleb pärida mõlemast andmebaasist kategooria kirjed nii, et sorteeritud oleks sama veeru järgi. Kui lihtsalt andmeid pärida ilma sorteerimata, siis nt Commence sorteerib vaikimisi välja järgi (ei pruugi olla nimi) tähestiku järjekorras, SQL aga kirjete loomise järjekorras (kuna aga kirjeid on loodud ka pärast esmast andmete importimist siis see ei katu tähestiku järjekorraga).

Commence'i sorteerimine nimi järgi:

```
category.SetSort "[ViewSort(""Name"",Ascending)]",0
```

Microsoft'i dokumentatsiooni järgi käib SQL käskude sorteerimine *order by* klauslit ning seejärel täpsustus kas kasvalt (A..Z, 0..9) või kahanevalt [6].

Ehk siis kui soovin otsida SQL serverist kõik töötajad ning sorteerida tulemus nime lahtri järgi siis teen järgmise päringu:

```
--OSUSR_KF2_EMPLOYEE on töötajate tabel SQL andmebaasis
select name, cmcid from OSUSR_KF2_EMPLOYEE order by name asc
```

Selleks et andmeid võrrelda leidis autor, et kõige mõistlikum on nii Commence'ist kui OutSystemsis küsitud andmed taandada JSON kujule. Näide autori profiili alusel:

```
[{"cmcid":"F:80011A01:9574D406","name":"Mart Potter"}]
```

Iga kirje kohta nii Commence'is kui OutSystemsis loeb autor andmed mällu ning kirjutakse JSON'i formaadis üles CMCID ja nimi. Commence'is nime ja ID küsimise kood on ära toodud Lisa 8 – Commence'is töötaja andmete küsimise VBS kood.

OutSystem'is (koodnimi „Jarvis“) samu andmeid küsides tuleb käivitada kood mis on ära toodud Lisa 9 - OutSystemsis töötaja andmete küsimise näide.

Tulemuseks on kahest erinevast allikast loetud kirjete Commence ID ja nime lahtri väärtuste jada JSON formaadis. Edasi või võrrelda neid json'i kirjeid omavahel:

```
if (commence_json <> jarvis_json) then
```

Võrdlus annab tulemuseks kirjed mis puuduvad või on vigased. Kood võimaldab ka vajadusel teha mingid lisategevused, nt administraatori teavitamise või automaatne andmete parandus. Sellise automaattesti võib käivitada koheselt peale esmast andmete

migratsiooni või nt mingi intervalliga, nt kord päevas. Selle kontrolli eeldus on, Commence'i pole antud kategooriast midagi ootel, mida OutSystemisse saata ja vastupidi, OutSystemis pole midagi ootel mida Commence'isse saata.

Kui nime ja id lahtri küsimine on lihtne, nt kuupäevadega on olukord veidi teistsugune. Commence rakenduse jaoks on märgitakse kuupäeva puudumine andmebaasis kui tühisõne („“), samas kui MSSQL salvestab selle kui „1900-01-01 00:00:00.000“. Kui originaalkujul andmeid võrrelda siis need pole võrdsed. Järelikult tuleb OutSystems (SQL) kuupäevi importides asendad „1900-01-01 00:00:00.000“ tühisõnega. Näiteks sünnipäeva pärimisel tuleb paranda andmed nii, et isikud kelle sünnipäeva pole märgitud saaks vasteks tühisõne.

```
Dateofbirth = FixDate(Employee_Recordset("Dateofbirth"))
```

Kus *Employee_Recordset("Dateofbirth")* on SQL väli ja FixDate on eraldiseisev funktsioon mille autor kirjutas:

```
Function FixDate(OldDate)
    If (OldDate = "1.01.1900") then
        FixDate = ""
    Else
        FixDate = OldDate
    End if
End Function
```

Ka jah/ei väärtuseid (boolean) töötlevad Commence ja OutSystems erinevalt. Selleks, et neid võrrelda tuleb need taandada binaarsele kujule:

```
If (Employee_Recordset("Isactive")) then
    Isactive = 1
else
    Isactive = 0
end if
```

Tingituna asjaolust, et Commence ei toeta Unicode'i siis on üks osa tervikluse tagamisest ka venekeelse teksti relatsioonilisele andmebaasile arusaadavaks tegemisele. Selleks defineeris autor ära väljad mis Commence'is sisaldavad (kas alati või teatud tingimustel) venekeelset teksti. Kui need tingimused oli täidetud siis kasutati järgmist VBS funktsioone mille algupärast varianti kirjeldati MoreTechTips veebilehel juba 2008 aastal [16]

```
temp=AlterCharset(temp,"windows-1257", "windows-1251")
```

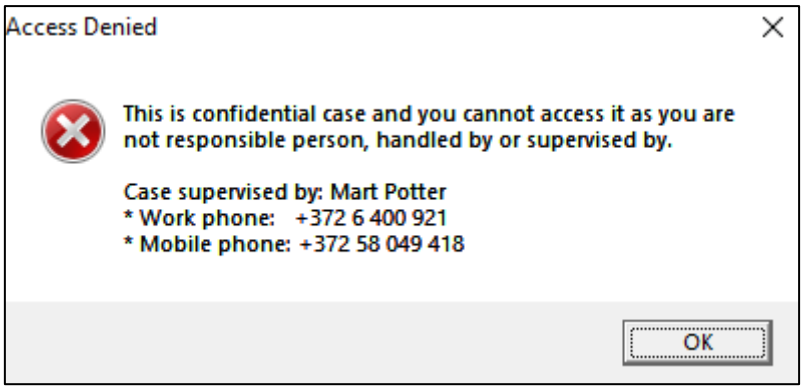
Kus *AlterCharset* on eraldi funktsioon mis on ära toodud Lisa 10 - AlterCharset VBS funktsioon.

4.3 Konfidentsiaalsus

Konfidentsiaalsus hõlmab sisuliselt salastatust, omandteavet ja/või privaatsust ja selle tähendus on sõltuvalt kontekstist kvalitatiivne ("on/ei ole konfidentsiaalne") või kvantitatiivne ("mil määral on/peab olema konfidentsiaalne") [27]. Ettevõtte jaoks on üldjuhul kõik andmed kõigile kasutajatele avalikud ning väljaspool ettevõtet neid jagada ei tohi – ehk siis ärisaladus on konfidentsiaalne. Erandina on välja toodud kodulehel avalikustavad andmed mis tulevad Commence'is ja / või Jarvisest. Ühelt poolt edastakse kodulehele info töötajate kohta, teisalt ka juba läbi saanud projektide kohta mida klient on lubanud avalikustada.

4.3.1 Konfidentsiaalsus Commence'is

Üldjuhul on kõik info avalik v.a. juhul kui see on piiratud. Ka piiramisega on mitu erinevat turvataset – osa infot piiratakse konkreetselt vormil (teatud välja on nähtavad teatud rollile), osa vorme saab avada vaid juhul kui kasutajale on antud sellele ligipääs (nt piiratud ligipääsuga projekt). Kui ligipääsus pole, siis saab kasutaja sellest teada nii nagu on näha Joonis 24 - Konfidentsiaalse näide Commence'is.

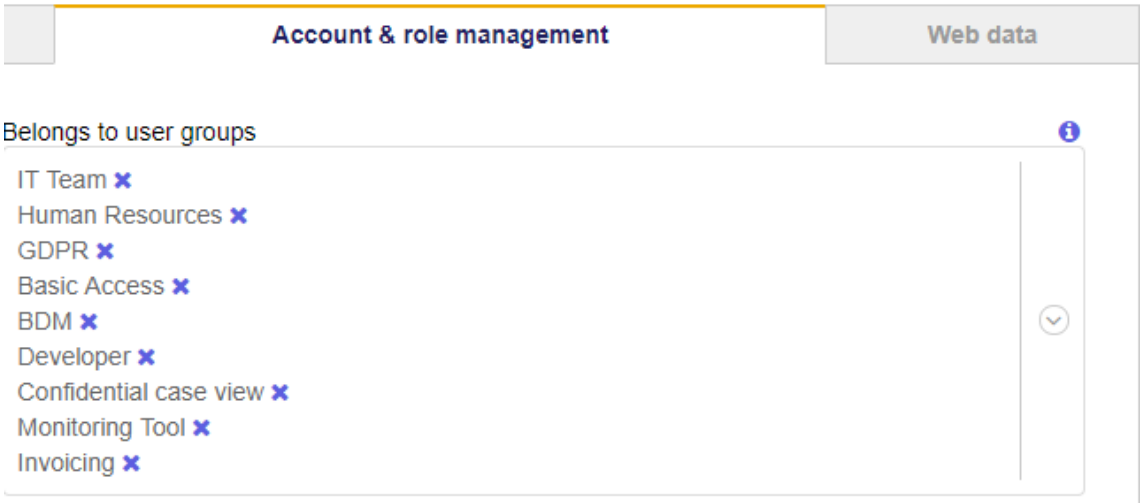


Joonis 24 - Konfidentsiaalse näide Commence'is, autori joonis

Selleks, et Commence saaks aru kellele mingit ligipääsu anda seotakse Commence'i andmebaasis kasutaja ühe olemiga töötajate tabelis. Töötaja olemi küljes on omakorda viide tema rollidele (seotud nt arveldusega). Vormi avamisel kontorollitakse automaatselt kes on kasutaja ja mis on tema rolli. Näiteks projekti vormi avamisel kontrollitakse kas projekt on konfidentsiaalne ja kui on siis kas kasutajal on projektile ligipääs. Kui projekt muudetakse konfidentsiaalseks või projekt on juba konfidentsiaalne siis teavitaks IT kasutajatuge kes määrab projektiga seotud failidele samad piirangud ka dokumendihaldussüsteemis. Samas käib see kasutaja valik andmebaasi pool ning seda saab vahetada ilma täiendava volituseta – tegemist on tõsise turvanõrkusega aga õnneks enamus kasutajaid ei tea sellest midagi.

4.3.1 Konfidentsiaalsus OutSystems'is

Nii nagu Commence on ka OutSystem'is teatud andmed piiratud kasutajatega – seda aga oluliselt paremini – nimelt saab ainult IT osakonna töötaja siduda kasutajakonto töötaja profiiliga ning lisada töötajaid teatud rolligruppidesse. Kasutaja saab kuuluda mitmesse gruppi (Joonis 25 - OutSystems kasutajaga soetud gruppide näide) ning iga grupi küljes on mingi roll (Tabel 6 - OutSystem'is kasutajale antud rollide näide), seega kasutaja saab teatud rolli läbi mingi õiguste grupi.



Joonis 25 - OutSystems kasutajaga soetud gruppide näide, autori joonis

Tabel 6 - OutSystem'is kasutajale antud rollide näide, autori tabel

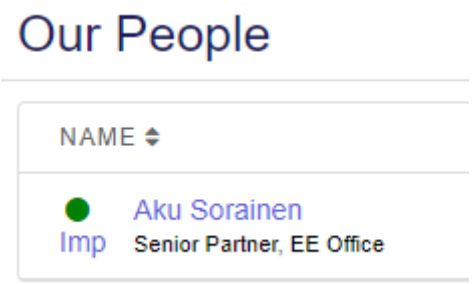
NAME	DESCRIPTION
Timecard_BilledTime_Write (from group 'IT Team')	Can view billed time on timecard form
Employee_SecNotes_Write (from group 'IT Team')	Can modify secure notes on employee form
Employee_PersonalData_Write (from group 'IT Team')	Can modify employee personal data
Position_IT (from group 'IT Team')	Position_IT

Edasi saab juba teatud elementidele määrata, et kui kasutajal on (läbi grupi) mingi roll, siis talle on see osa kättesaadav (nähtav, muudetav). Joonis 26 - OutSystems rollipõhine info kuvamine illusteerib, kuidas teatud vormi osa on nähtav vaid IT rollile. Pannes selle info kokku Tabeliga kuus on näha, et autor sai rolli „Position IT“ läbi grupi „IT Team“.

Joonis 26 - OutSystems rollipõhine info kuvamine, autori joonis

4.3.2 Konfidentsiaaluse testimine

Nii Commence'is kui Outsystem'is saab kasutada impersoneerimist. Vahe on aga selles, et kui Commence'is saab teha seda igaüks kes selle funktsionaalsuse üles leiab, siis Outsystem'is on see seotud rangelt vaid arendaja rolliga isik (nt isegi kasutajatugi seda teha ei saa). Volitatud isikule tekib töötaja nime kõrvale link „Imp" nii nagu on näha Joonis 27 - impersoneerimise näide OutSystemsis, autori joonis.



Joonis 27 - impersoneerimise näide OutSystemsis, autori joonis.

Kui Commence'i ei jää antud tegevusest ka mingid jälge, siis OuySystemsi pool tehakse automaatselt andmebaasi logikirjed. Logikirjete näide on ära toodud Tabel 7 - OutSystemsi impersoneerimise tabeli näidis.

Tabel 7 - OutSystemsi impersoneerimise tabeli näidis, autori tabel

ID	PREVIOUSUSERID	IMPERSIONATEDUSERID	LOGINDATETIME
534	333	11	14.05.2019 21:07
533	11	333	14.05.2019 20:49
532	12	99	14.05.2019 12:16
531	36	17	14.05.2019 11:34

See on juba hea algus, aga autor leiab, et arenguruumi veel on – nimelt kuna see tabel pole krüpteeritud siis nt andmebaasi haldajad kirjeid muuta ja ka kustutata. Selleks on mõistlik tulevikus ära kasutada siduda juba toiminud sündmused põhjusliku ehk kausaalse seose abil. Tulemuseks on lingitud tembeldamine ehk Tabel 7 näitel arvutakse kirje 534 räsi nii, et see sõltub kirje 533 räsist, mis omakorda sõltub 532 räsist jne. Seega ei saaks peale kirjet 534 tekitada kirjet mis oleks ajaliselt enne kirjet 533. Lingitud tembeldamist kajastab autor ka oma ettekandes „Ajatempel" aine Küberturbe arhitektuur raames [68].

4.4 Andmete turvalisuse kokkuvõte

Autor leidis, et konfidentsiaalsust saab tänapäevaste veebitehnoloogiatega oluliselt paremini realiseerida kui Commence, seda siis eelkõige läbipaistava rollide haldusega. Autor hinnangul saab ka käideldavust OutSystemis tagada oluliselt paremini kui Commence'is eelkõige sellega, et andmed saab siduda selle looja ja muutjaga. Andmete tervikluse tagamine on autori hinnangul kogu projekti võtmeküsimus ning selle lahendamiseks realiseeris autor automaattesti prototüübid mida on praktikas kas juba korduvalt kasutanud kuid mida peab kindaslti edasi arendama.

5 Kokkuvõte

Diplomitöö eesmärk oli kirjeldada parimad praktikad ning pidepunktid millest lähtudes oleks täidetud eelkõige andmete terviklus ja käideldavus Advokaadibüroo Sorainen AS vaates uue platvormi evitamisel. Töö käigus kirjeldas autor olemasolevat arhitektuuri ning selle moderniseerimisega seotud väljakutseid keskendudes olemasolevate liidestuste üleviimisele ning vaadates lähemalt pärandsüsteemi ja uue platvormi vahelise infovahetust.

Diplomitöö tulemusena arendas autor välja ettevõtte jaoks vajalikud automaattesti prototüübid millega saab kontrollida kas andmed on kadudeta üle ja millised on erinevused ning vajadusel pärandsüsteemi aluseks võttes andmeid automaatselt korrigeerida.

Autor leiab, et loodud algoritmid on väga head vigade leidmiseks ning kõrvaldamiseks. Antud diplomitöö raames kirjutas autor vaid automaattestide prototüübid ning selleks, et andmete terviklust kõikide kategooriate, veergude ning väljade tasemel kontrollida tuleb veel palju aega kulutada. Autor võtab need testid kasutusel esialgu kõige kriitilisemat olemite jaoks Tagantjärgi tark olles oleks pidanud sellise valideerimisprotsessi ja algoritmi loomisega tegelema projekti varasem järgus.

Edaspidi on plaanis 2019 aasta lõpuks kõik Commence'i andmed OutSystemisse üle viia ning andmete eelkõige andmete ülekandmisel veel rohkem rõhku panna tervikluse tagamisele. Autor on kindel, et mida kiiremini saab Commence'is funktsionaaluse kinni keerata seda tõhusamalt saab kogu süsteemi turvalisust (nii käideldavust, konfidentsiaalsust kui turvalisust) juba OutSystem'i pool tagada.

6 Kasutatud kirjandus

1 Meist – *Sorainen*
https://www.sorainen.com/et/meist/ (18.03.2019)

(artikkel veebist)

2 Data Integrity Strategies for Migrating Data from Legacy Systems - *Ivan Soto*
http://www.ivtnetwork.com/article/data-integrity-strategies-migrating-data-legacy-systems (17.03.2019)
(artikkel veebist)

3 @@rowcount (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/functions/rowcount-transact-sql?view=sql-server-2017 (19.03.2019)
(artikkel veebist)

4 VBScript — Using error handling - *Stack Overflow*
https://stackoverflow.com/questions/157747/vbscript-using-error-handling (19.03.2019)
(artikkel veebist)

5 COUNT (Transact-SQL) – *Microsoft SQL Docs*
https://docs.microsoft.com/en-us/sql/t-sql/functions/count-transact-sql?view=sql-server-2017 (22.03.2019)
(artikkel veebist)

6 SELECT - ORDER BY Clause (Transact-SQL) – Microsoft SQL Docs
https://docs.microsoft.com/en-us/sql/t-sql/queries/select-order-by-clause-transact-sql?view=sql-server-2017 (22.03.2019)
(artikkel veebist)

7 E-Teadmik - *Vallaste*
http://vallaste.ee/ (25.03.2019)
(artikkel veebist)

8 Scrum - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/2148-valearendus (02.04.2019)
(artikkel veebist)

9 What Is Low-Code?- *OutsSytems*
https://www.outsystems.com/blog/what-is-low-code.html (25.03.2019)
(artikkel veebist)

10 CSS - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/8917-css-1, 02.04.2019
(artikkel veebist)

11 Foreign Key - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/8150-foreign-key (25.03.2019)
(artikkel veebist)

12 Use the Microsoft Graph API - *Microsoft Graph*
https://docs.microsoft.com/en-us/graph/use-the-api (25.03.2019)
(artikkel veebist)

13 API - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/3088-api, 25.03.2019
(artikkel veebist)

14 PowerShell Ülevaade - *Urmas Tamm, Alo Peets*
https://courses.cs.ut.ee/MTAT.03.005/2014_fall/uploads/Main/PSylevaade2014.pdf, 25.03.2019
(artikkel veebist)

15 Terviklus – *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/894-terviklus, 02.04.2019
(artikkel veebis)

16 Convert String to Bytes using a character set (and Vice versa) – *10.23.2008, MoreTechTips*
http://www.moretechtips.net/2008/10/convert-string-to-bytes-and-vice-versa.html (26.03.2019)
(artikkel veebist)

17 Märgistuskeel - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/6699, 02.04.2019
(artikkel veebist)

18 Autentsus - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/381-autentsus, 02.04.2019
(artikkel veebist)

19 Salgamatus - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/446-salgamatus, 02.04.2019
(artikkel veebist)

20 Json - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/10850-json, 02.04.2019
(artikkel veebist)

21 Data Migration Testing Techniques to Migrate Data Successfully – *iCEDQ insight,*
https://icedq.com/data-migration/data-migration-testing-techniques-to-migrate-data-successfully, 02.04.2019
(artikel veebis)

22 The Data Migration Process & the Potential Risks - iCEDQ insight,
https://icedq.com/data-migration/the-data-migration-process-and-the-potential-risks, 02.04.2019
(artikel veebis)

23 The 4 layaer canvas – *OutSystems*
https://success.outsystems.com/Support/Enterprise_Customers/Maintenance_and_Operations/Designing_the_architecture_of_your_OutSystems_applications/01_The_4_Layer_Canvas, 03.04.2019
(artikkel veebist)

24 Validating your application architecture – *OutSystems*
https://success.outsystems.com/Support/Enterprise_Customers/Maintenance_and_Operations/Designing_the_architecture_of_your_OutSystems_applications/03

_Validating_your_application_architecture, 03.04.2019
(artikkel veebist)

25 ITIL - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/3243-itil, 07.04.2019
(artikkel veebist)

26 Käidelavus - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/29-kaideldavus, 19.04.2019
(artikkel veebis)

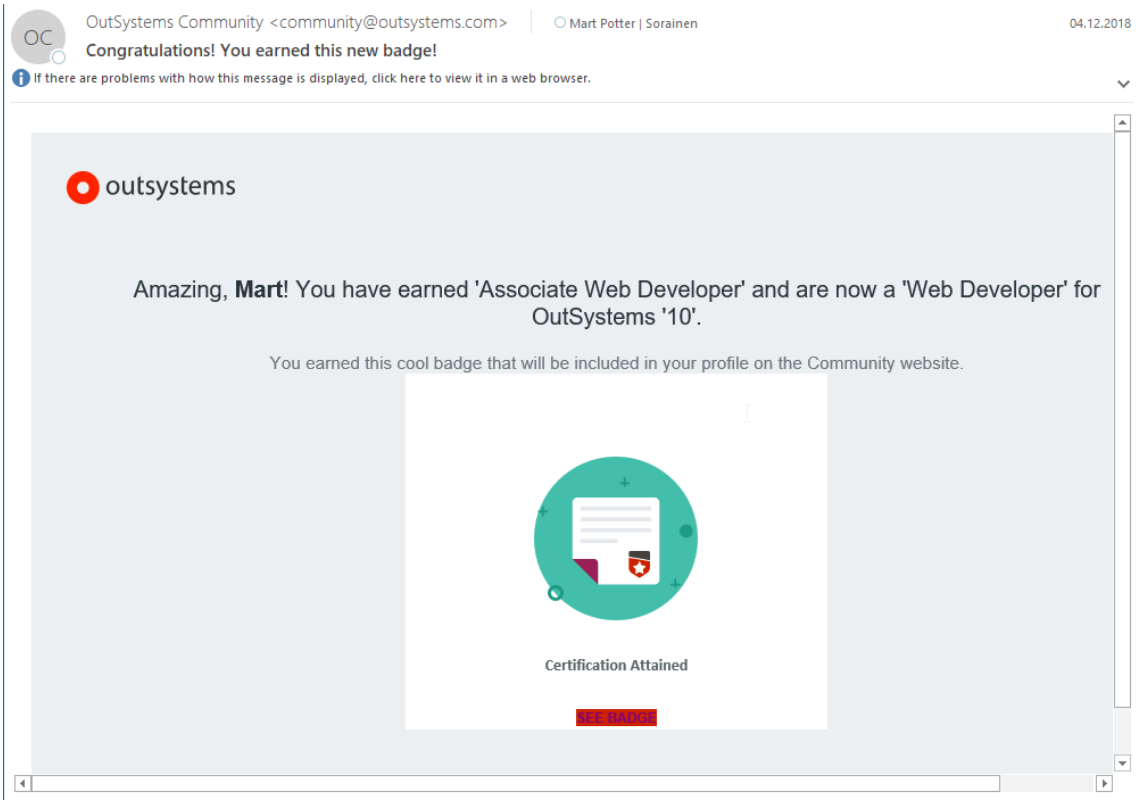
27 Konfidentsiaalsus - *Andmekaitse ja infoturbe leksikon*
https://akit.cyber.ee/term/416-konfidentsiaalsus, 28.04.2019
(artikkel veebis)

28 CIA triad - *TechTarget*
https://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA, 04.05.2019

29 Ajatempel – *autori ettekanne aine Küberturbe arhitektuur raames*
https://docs.google.com/presentation/d/1rc5hsWTvUgi3k1Q6zyu4Mmn7RoByae-uD8zGt7OdoK8/edit?usp=sharing, 14.05.2019
(ettekanne)

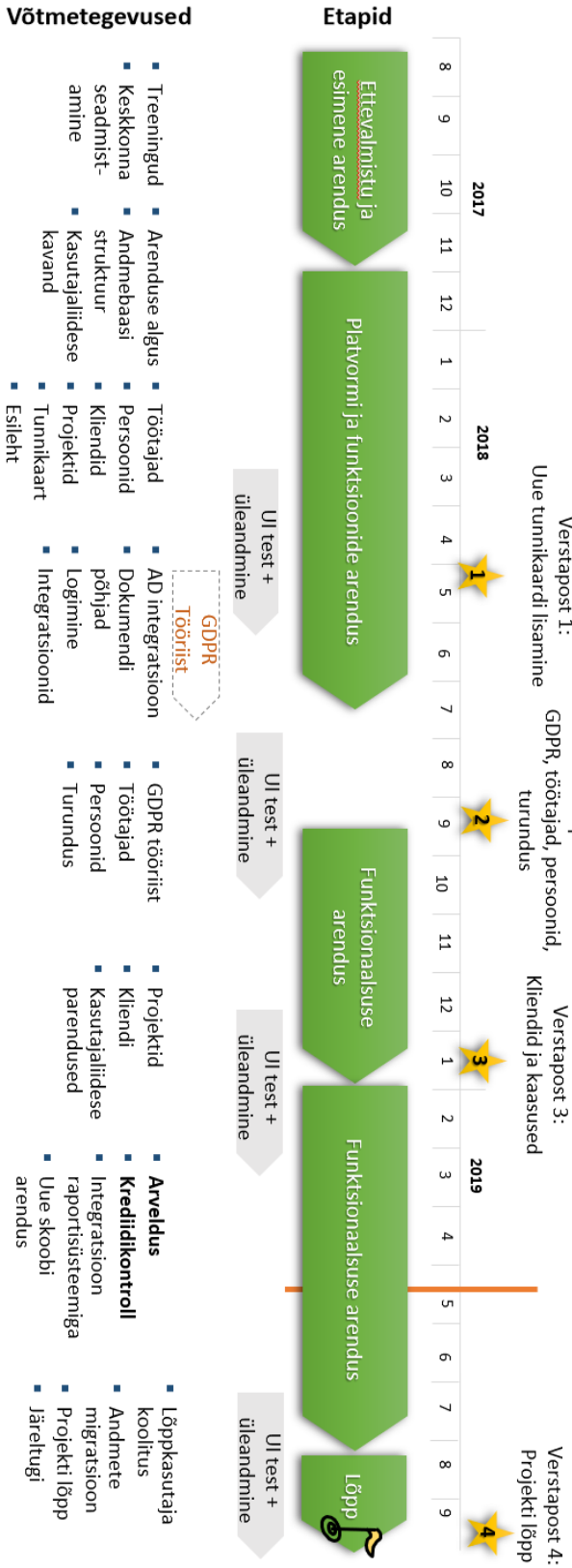
7 Lisad

Lisa 1 Associate Web Developer sertifikaat



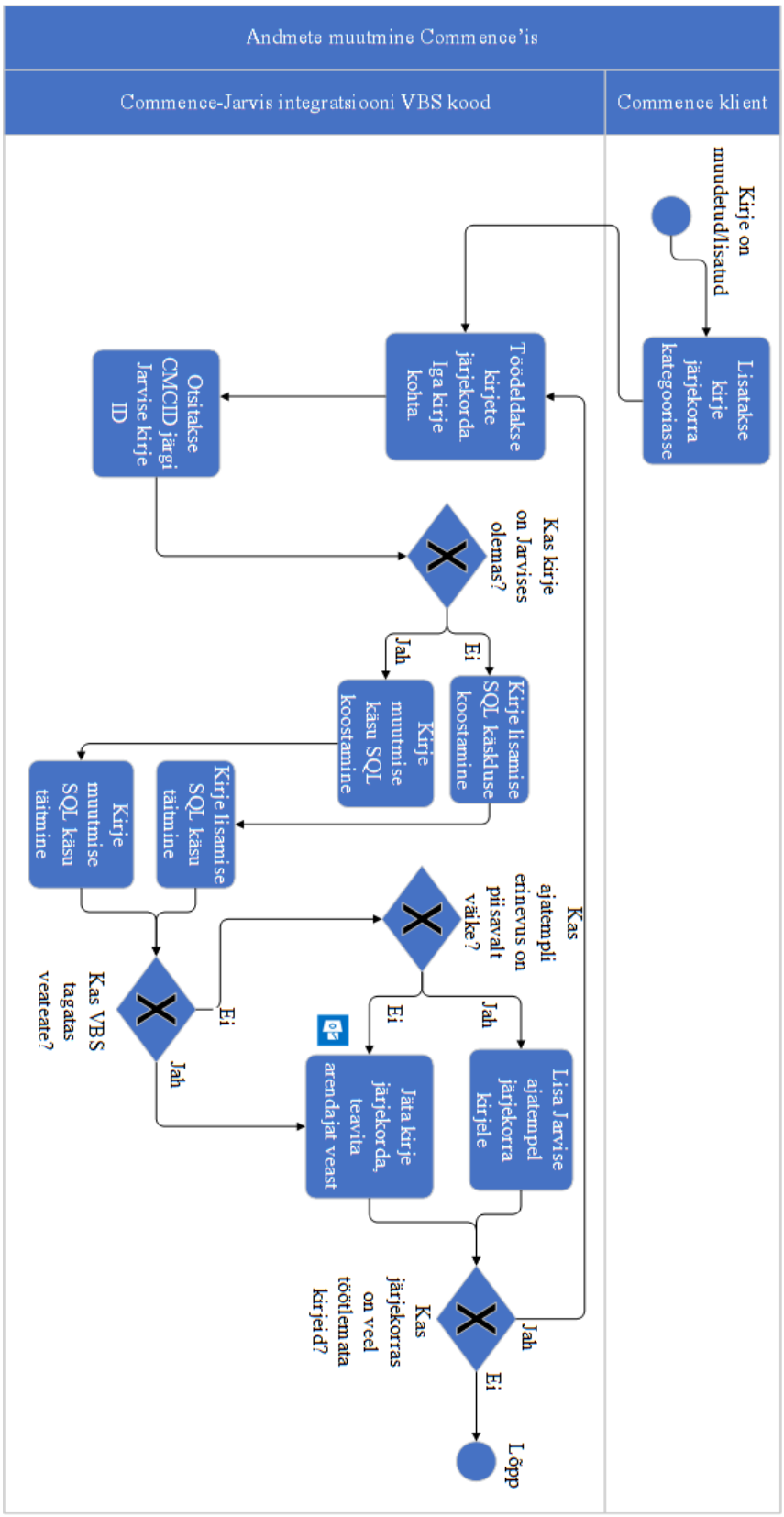
Joonis 28 - Associate Web Developer sertifikaat

Lisa 2 - Projekti plaan



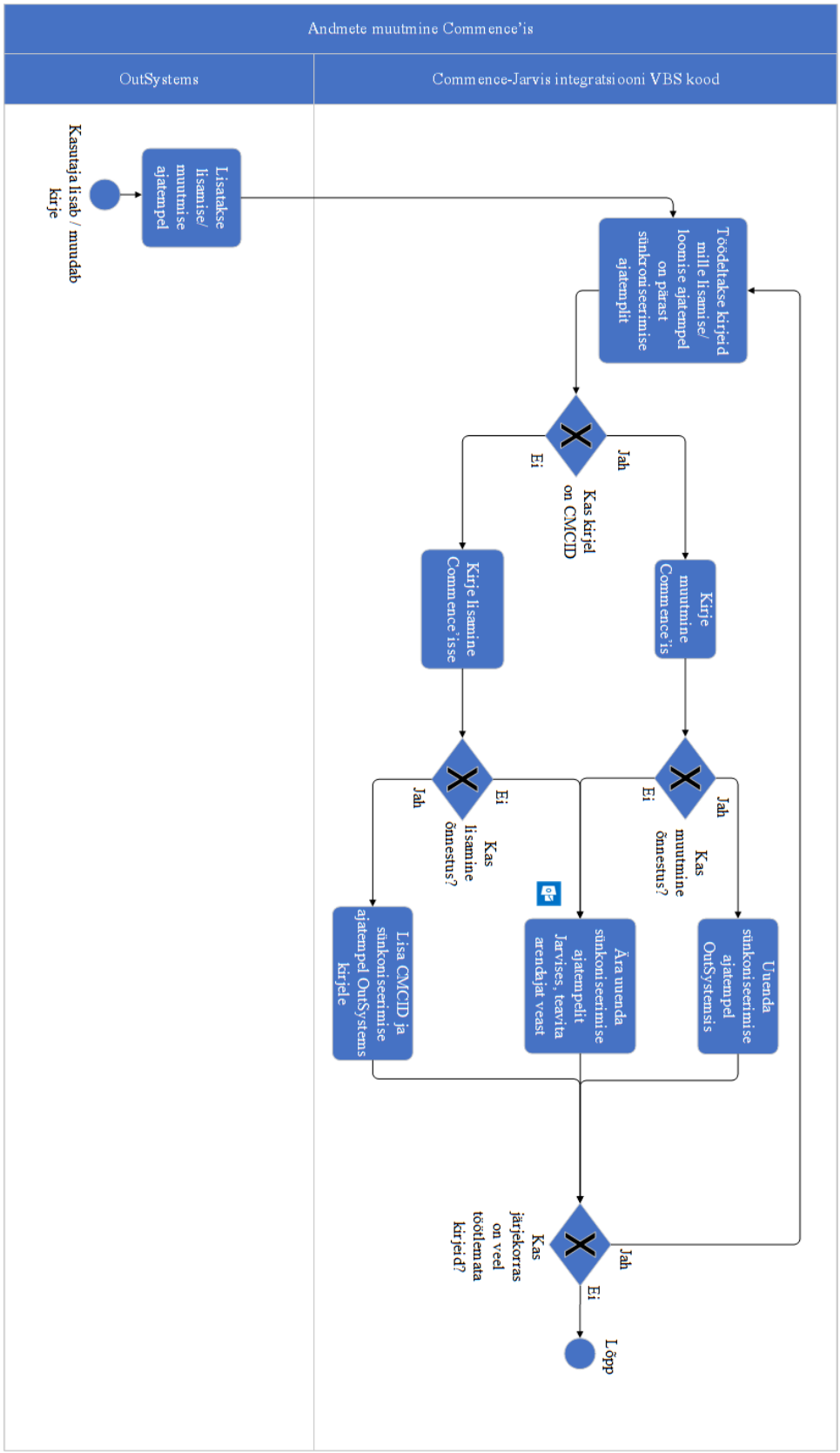
Joonis 29 - Projektiplaan, projektijuhis joonis, autori tõlge

Lisa 3 - Commence'i poolt andemete muutmise töövoog



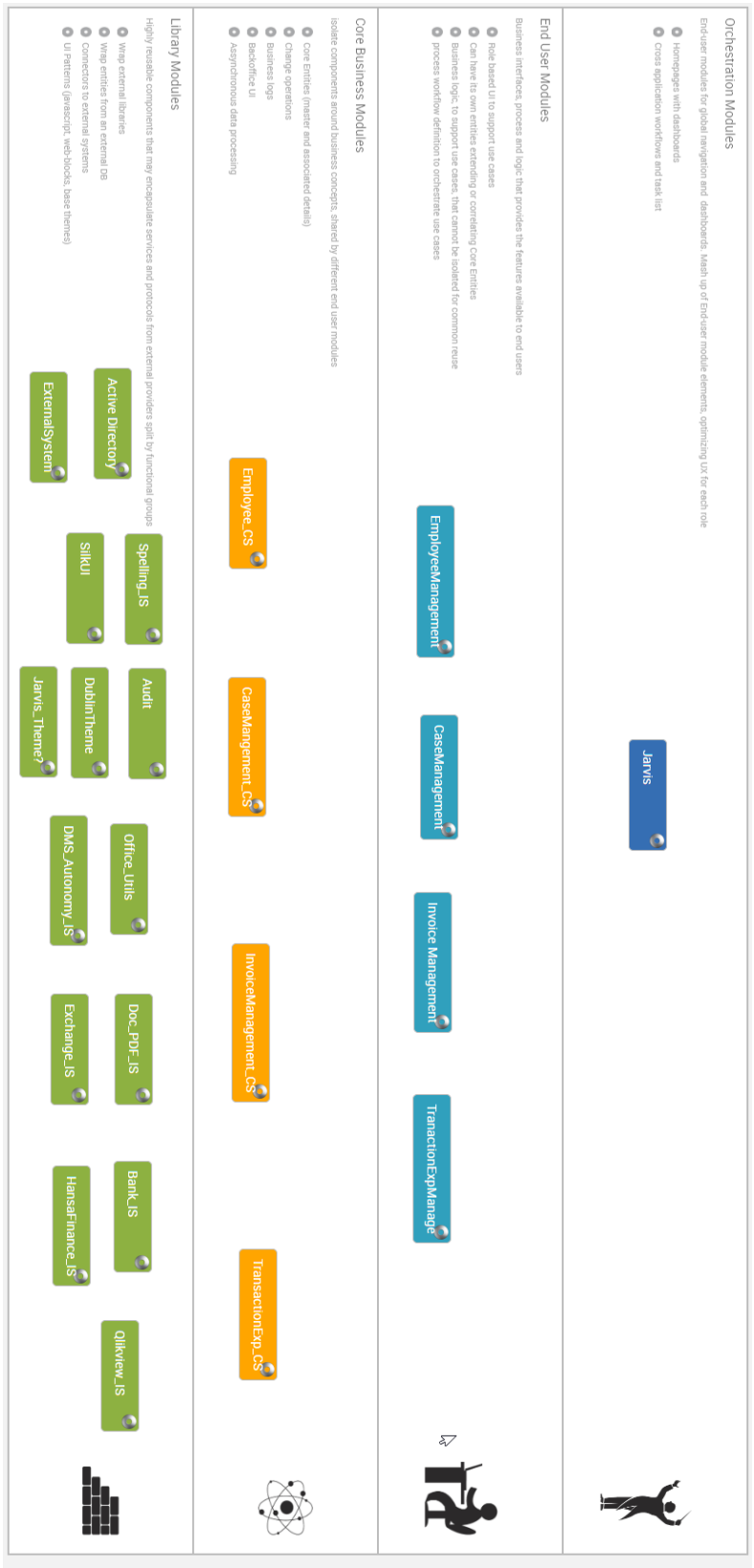
Joonis 30 - Commence'i poolt andemete muutmise töövoog, autori joonis

Lisa 4 – OutSystemi poolt andmete muutmise töövoog



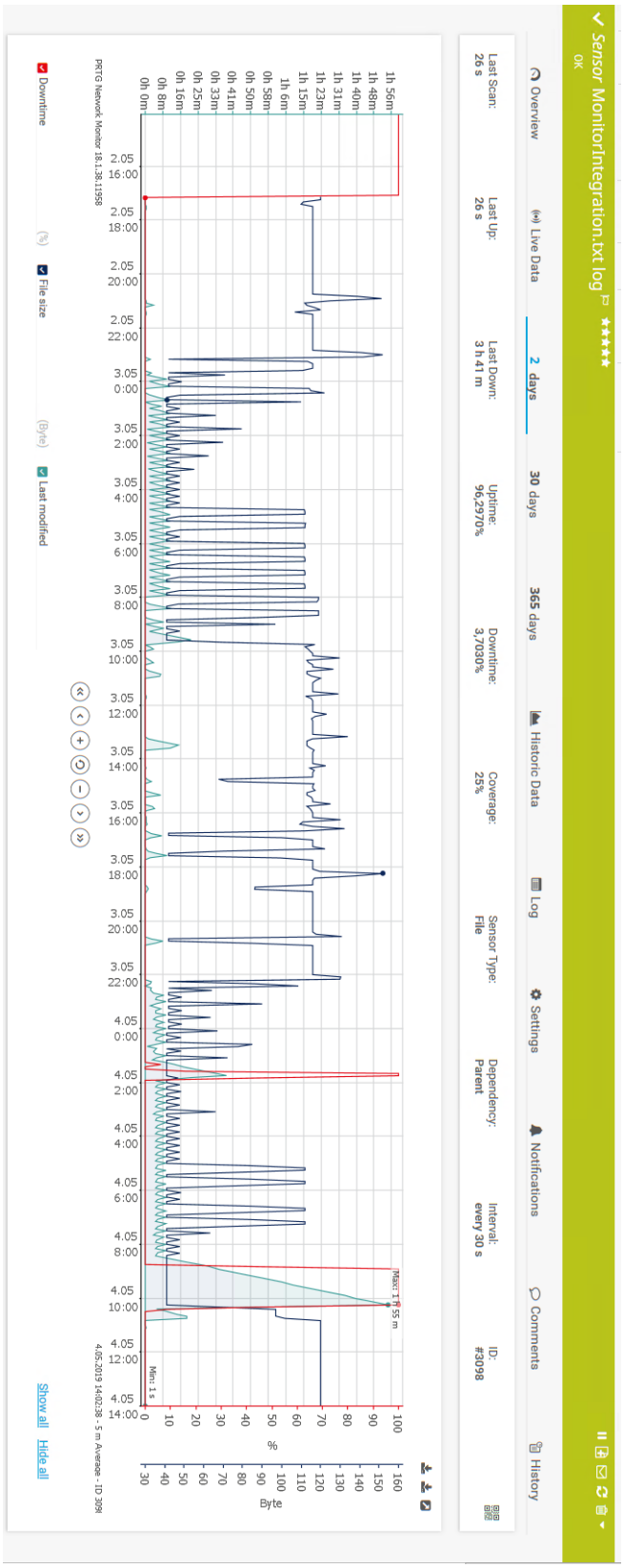
Joonis 31 - OutSystemi poolt andmete muutmise töövoog, autori joonis

Lisa 5 – Neljakihilise arhitektuuri hetkeseis



Joonis 32 - Neljakihilise arhitektuuri hetkeseis, autori joonis

Lisa 6 – PRTG sensori MonitorIntegration.txt log graafik



Joonis 33 - PRTG sensori MonitorIntegration.txt log graafik, autori joonis

Lisa 7 – ExecuteSQL VBS funktsioon

```
Function ExecSQL(ByVal sqlstr, ByVal action)
        On Error Resume Next
        objTextFile.WriteLine(Time() & " | " & sqlstr)
        cmd.CommandText = sqlstr
        If action=0 Then
                Set rss=cmd.Execute(strSQL)
                ExecSQL=rss.Fields(0)
        End If
        If action=1 Then
                cmd.Execute "strSQL"
                ExecSQL=0
        End If
        If Err.Number <> 0 Then
                objError.WriteLine(Time() & "[Err #" & Err.Number & "] | " & sqlstr)
                Call sendHTMLMail ("mart.potter@sorainen.com", "ExecSQL", "[Err #" &
                Err.Description & "] | " & sqlstr)
        Err.Clear
        End If
End Function
```

Lisa 8 – Commence'is töötaja andmete küsimise VBS kood

```
For i=1 to category.RowCount()
	Set qrs = category.GetQueryRowSet(1,0)
	if (i = 1) then
		commence_json = "["
	else
		commence_json = commence_json & ","
	end if

	Set qrs = category.GetQueryRowSet(1,0)
	name=qrs.GetRowValue(0,qrs.GetColumnIndex("Name", 0),0)
	cmcid=qrs.GetRowID(0,0)

	commence_json = commence_json & "{" & q("cmcid") & ":" & q(cmcid) & "," & q("name") & ":" & q(name) & "}"
Next
commence_json = commence_json & "]"
```

Kus funktsioon *q* paneb sõnele jutumärgid ümber:

```
Function q(input)
	q = chr(34) & input & chr(34)
End Function
```

Lisa 9 - OutSystemsis töötaja andmete küsimise näide

```
Set Employee_Recordset = CreateObject("ADODB.Recordset")
Do While NOT Employee_Recordset.Eof
        Cmcid = Employee_Recordset("Cmcid")
        name = Employee_Recordset("EmployeeName")
        if (i = 1) then
                jarvis_json = "["
        else
                jarvis_json = jarvis_json & ","
        end if
        jarvis_json = jarvis_json & "{" & q("cmcid") & ":" & q(cmcid) & "," & q("name") & ":"
& q(name) & "}"

        Employee_Recordset.movenext

Loop
jarvis_json = jarvis_json & "]"
```

Lisa 10 - AlterCharset VBS funktsioon

```
Function AlterCharset(Str, FromCharset, ToCharset)
        Dim Bytes
        Bytes = StringToBytes(Str, FromCharset)
        AlterCharset = BytesToString(Bytes, ToCharset)
```

```
End Function
```

Kus StringToBytes ja BytesToString on eraldi funktsioonid

```
' accept a string and convert it to Bytes array in the selected Charset
Function StringToBytes(ByVal Str, ByVal Charset)
        Set stream = CreateObject("ADODB.Stream")
        Stream.Type = 2
        Stream.Charset = Charset
        Stream.Open
        Stream.WriteText Str
        Stream.Flush
        Stream.Position = 0
        ' rewind stream and read Bytes
        Stream.Type = 1
        StringToBytes= Stream.Read
        Stream.Close
        Set Stream = Nothing
End Function
```

```
' accept Bytes array and convert it to a string using the selected charset
Function BytesToString(ByVal Bytes, ByVal Charset)
        Set stream = CreateObject("ADODB.Stream")
        Stream.Charset = Charset
        Stream.Type = 1
        Stream.Open
        Stream.Write Bytes
        Stream.Flush
        Stream.Position = 0
'       rewind stream and read text
        Stream.Type = 2
        BytesToString= Stream.ReadText
        Stream.Close
        Set Stream = Nothing
End Function
```