TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatika

Magnus Muru 184912IAIB

TOOTED KASSASÜSTEEMI INFOSÜSTEEMI VERSIOON 2.0

Bakalaureusetöö

Juhendaja
Gert Kanter
PhD

Tallinn 2022

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:	Magnus Muru
Kuupäev:	30.05.2022

Annotatsioon

Coop Eesti Keskühistu eesmärk on toetada kõigi 19 Eestis asuva ühistu tööd. Üks peamistest tugiteenustest, mida Keskühistu pakub, on kaupade logistika ning sellega seotud andmevahetus. Kõike seda haldabki täna Tooted Kassasüsteemi (TKS) süsteem. Kuigi tänane lahendus tagab andmeliikumise Keskühistu ning kassasüsteemide vahel, siis praeguseks on see muutunud pärand-tarkvaraks ja pärsib nii enda kui teiste süsteemide jätkusuutlikust.

Selle lõputöö eesmärk oli luua uus TKS 2.0 süsteem mis tagaks tulevikukindla, läbipaistva ning jõudliku lahenduse vahetamaks välja hetkel kasutusel oleva verisooni. Selle jaoks oli vaja luua süsteem mis võimaldab luua modulaarset andmestruktuuri ning mille põhjal on võimalik genereerida ka tänane andmemudel kuni kassasüsteemide TKS 2.0 andmestruktuuri ülevõtmiseni. Lõputöös käsitletakse tänase süsteemi piiranguid ja sellest lähtuvalt tehtud muudatusi tarkvaraarhitektuuris ning rakenduses. Töö tulemustest leidus kinnitust, et võrreldes vana süsteemiga kiirendab uus rakendus andmevahetust keskhaldussüsteemide ja kassasüsteemide vahel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 7 peatükki, 6 joonist, 1 tabel.

Abstract

TOOTED KASSASÜSTEEMI INFORMATION SYSTEM VERSION 2.0

Coop Eesti Keskühistu goal is to support the 19 cooperatives that are located in Estonia. One of the primary support services that central cooperative offers is logistics and the related data exchange, which is currently being maintained by Tooted Kassasüsteemi (TKS) system. Although, today's solution ensures data transfer between the central cooperative and cash register systems, it has over time turned into legacy software, inhibiting system throughput.

The goal of this thesis was to create a new TKS 2.0 system, that would ensure a future-proof, transparent and performant solution that would replace the current. For this case, there was a need for a system that would support a modular data structure which in turn would allow to generate the sufficient data model for cash register systems until they accept TKS 2.0 data model. The thesis deals with the current system's limitations and the according changes taken in software architecture and application. The performance analysis comparison between the old and the new system shows how the new application improves data transfer speed.

The thesis is written in Estonian and is 24 pages long, including 7 chapters, 6 figures, and 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*
Azure	Microsofti pilveteenus
Blazor	C# ja HTML baasil vabavaraline veebiraamistik
C#	*C Sharp*
DevOps	*Development Operations*
EF	*Entity Framework*
HTML	*HyperText Markup Language*
IIS	*Internet Information Services*
JWT	*JSON Web Token*
LDAP	*Lightweight Directory Access Protocol*
LINQ	*Language Integrated Query*
ORM	*Object–relational mapping*
RAM	*Reliability, Availability, and Maintainability*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
TKS	Tooted Kassasüsteemi
XML	*Extensible Markup Language*

Sisukord

Jooniste loetelu **vii**

Tabelite loetelu **viii**

1 Sissejuhatus **1**

2 Olemasolev rakendus **2**

2.1 Liidestus keskhaldussüsteem Lexiga 2

2.2 TKS ülesehitus 3

2.2.1 Olemasolev infrastruktuur 3

2.3 TKS süsteemi üleviimise metoodika 4

3 Tehnoloogiad **5**

3.1 .NET platform 5

3.1.1 Entity Framework Core 6

3.1.2 Blazor 6

3.1.3 Serilog 7

3.2 Docker 7

3.3 Elasticsearch & Kibana 7

3.4 SQL Server 7

3.5 IIS 8

3.6 Azure DevOps 8

4 Arhitektuur **9**

4.1 Sortimendiinfo töövoog 12

4.1.1 Toote-, hinna- ja kampaaniainfo töövoog 12

4.2 Liidestamine andmebaasiga 13

4.3 Veebihaldusliides 15

4.4 Logimine ja monitooring 15

4.5 Autentimine ja autoriseerimine 16

4.6 Teenuse levitamine 16

5 Kasutamine **17**

5.1 Veebiliidese kasutamine 17

5.2 TKS 2.0 kasutamine 17

5.3 Proksi kasutamine 19

6 Tulemuste analüüs **21**
6.1 Töökindlus 21
6.1.1 Liiasus 21
6.1.2 Testimine 22
6.2 Käideldavus 23
6.2.1 Modulaarsus 23
6.3 Hooldatavus 23
6.3.1 Diagnostika 24

7 Kokkuvõte **25**

Kasutatud kirjandus **26**

Lisad **28**

Lisa 1 - Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks **28**

Lisa 2 - Administraatori vaade kasutajahalduse veebiliideses **29**

Joonistе loetelu

1 Süsteemiprojektide sõltuvused 10
2 Süsteemi üldtöövoog 11
3 Protsess väljundsõnumite tekkeks 13
4 Andmeühenduse töövoog 14
5 Autentimise töövoog 18
6 Proksi andmepakkide loomine 20

Tabelite loetelu

1 Ajakulu vana süsteemi ja TKS 2.0 süsteemi andmete loomisel. 22

1. Sissejuhatus

Coop Eesti Keskühistu pakub tuge üheksateistkümnele Eestis asuvale Coop ühistule. Üheks oluliseimaks on kaupade logistika ning nendega seotud andmevahetus, mille keskseks teenuseks on Keskühistu ja ühistute vahel Tooted Kassasüsteemi infosüsteem. Selle süsteemi halduses on kogu Keskühisu toodetega seotud keskhaldussüsteemide andmete liigutamine erinevatele ühistute poolt kasutatavatele kassasüsteemidesse. TKS süsteemi pikaajalisel kasutamisel on selgunud süstemaatilised kitsaskohad mida tänane süsteem ei suuda täita, nagu limiteeritud käideldatavus ning hooldatus.

Praeguse lahenduse jätkusuutlikuks muutmine algas 2019. aastal ning esialgse ärianalüüsi raames selgus koheselt, et varasem süsteem on muutunud pärandtarkvaraks ja vajab täielikult uuendamist. Selle alusel valmis esmane idee luua TKS 2.0 kesksüsteem, mis oleks tulevikukindel ning valmis erinevate ärivajadustega kaasas käima. Analüüsi põhjal hakati looma esialgset süsteemi mille andmestruktuur ning andmevahetus oleks aluseks järgnevatele TKS versioonidele, ning samaaegselt oleks võimeline tänaste kassasüsteemide vajalikuks toimimiseks andmestruktuuri. Töö käigus jaotati need süsteemiülesanded kaheks seotud projektiks:

- TKS 2.0 kesksüsteem
- TKS proksi

Lõputöö keskendub TKS 2.0 kesksüsteemi (edaspidi TKS 2.0) loomisele, mille puhul autor on üksikasjalikult panustanud süsteemilahenduste analüüsi, planeerimisse, ning arendamisesse. Samuti tuuakse välja TKS proksi süsteem, mille loomisel on kaasa löönud ka teised Coop Keskühistu arendajad ja kuhu autor panustab liidestamisesse TKS 2.0 süsteemiga.

2. Olemasolev rakendus

Tänane TKS lahendus on ehitatud Oracle andmebaasile. Selle tarbeks on rajatud regulaarsed andmevahetused mis erinevate andmete ja andmemuudatuste abil kopeerivad vajaminevad andmed üks-ühele ümber Lexi keskhaldusbaasist Oracle serveris asuvasse andmebaasi. Antud peatükis tuuakse välja, millised on selle lahenduse puudused ja eelised võrdluses lõputöös loodud lahendusega.

2.1 Liidestus keskhaldussüsteem Lexiga

Coop logistikaahela erinevate osade toetuseks on kasutusel mitmed tarkvarasüsteemid erinevate andmemudelitega. Nendevahelise andmevahetuse tagamiseks on oluline tagada andmete keskne haldamine ning ühtne andmete seis. Coopis täidab seda ülesannet keskhaldussüsteem Lexi, mille krütpeeritud liidesed koguvad ning täiendavad erinevate süsteemide andmeid. Peale REST-põhiste APIde on Lexil lisaks sõnumijärjekord, mida kasutatakse Lexi andmeid tarbivatele süsteemidele info edastamiseks. Selle abil kantakse üle erinevaid andmeid, mille puhul TKSi jaoks oluliseimad võib kategoriseerida järgnevatesse gruppidesse:

- sortimendiinfo
- tooteinfo
- hinnainfo
- kampaaniainfo

Täielikult saab keskhaldussüsteemist kätte TKS sortimendi-, toote- ning hinnainfo ja ka osaliselt kampaaniainfo, mida TKS täiendab kampaaniainfo haldussüsteemist. Kuna osad kriitilised andmevahetusprotsessid keskhaldusüsteemis toimivad sünkroonselt, on ka TKS üles ehitatud sünkroonseks andmevahetuseks mis toimib *cron* töö raames. TKS töö algab sõltumata andmevahetuse lõppemisest, arvestades umbkaudset andmevahetuse lõppemist koos puhverajaga mis võib muutuda kriitiliseks kui toimuvad suuremad andmemuudatused näiteks seoses hooajaliste kaupade vahetamisega. Antud süsteemi eelis on andmete üksus TKSi ja keskhaldussüsteemide vahel, kuid nõuab korduvalt rohkem riistvaralist lisaresurssi andmete dubleerimise tõttu.

2.2 TKS ülesehitus

TKS on rajatud Oracle andmebaasis toimuvate regulaarsete tööde peale, mis vastavalt relevantsete andmete põhjal loob andmepakke, mis omakorda koosnevad keskhaldussüsteemist tulevatest neljast infoliigist. Selle jaoks kopeeritakse üle sõnumijärjekorra andmemuudatused, mis on TKS toimimise jaoks olulised, ning luuakse vastava kassasüsteemi jaoks sobiv andmemudel. Antud andmemudel on üks monoliitne pakk mille loomiseks toimuvad järgnevad protsessid:

1. TKS loob vahetabelid kuhu koondatakse kogu info järgmisel päeval saadetavate TKS pakkide jaoks
2. TKS loob XML mudelil baseeruvad pakid mis talletatakse andmebaasi kuni saatmiseni
3. TKS saadab üle SOAP teenuse välja igale kassasüsteemile relevantse informatsiooniga infopaki ning kustutab need oma süsteemist

Nende tööde suurim piirang on läbipaistvuse puudumine - limiteeritud logimine ning puudulik erinditöötlus. Seetõttu ei ole võimalik tuvastada eduka või ebaeduka andmevahetuse toimumist kassasüsteemi ja TKSi vahel. Lisaks on puudulik andmete korrektsuse automaatne kontroll. Erindite esinemisel nõuab selle lahendamine käsitsi tuvastamist, uuesti sisestamist, ning puuduolevate andmepakkide taasloomist ja saatmist. Parandamisel tuleb luua uuesti terve andmemudel, kuna kassasüsteemide liidestus ei võimalda olemasoleva süsteemi puhul andmete osalist ülekannet. Nende toimingute teostamisega kaasneb suure andmekogusega vahetabelite loomine andmebaasis, mille puhul on tegemist väga arvutusvõimsuselt ressursimahuka toiminguga mida vigade tekkimisel korrata.

2.2.1 Olemasolev infrastruktuur

Lõputöö raames täiendati Coop Eesti Keskühistu olemasolevaid keskhaldusinfosüsteeme, et vahetada välja olemasolev TKS lahendus. Selle otstarbeks oli vaja laiendada ning omavahel ühendada ja integreerida järgnevaid süsteeme uue TKS 2.0 lahendusega:

- Lexi MessageQueue;
- Lexi REST-baasil API lahendused;

Täiendavalt loodi TKSi andmevahetuse toetamiseks uued andmebaasid ning virtuaalserverid. Riistvaravahendid, mida uued süsteemid kasutavad on järgnevad:

- virtuaalserverid
  - Lexi test- ja tootekeskkonnad
  - TKS test- ja tootekeskkonnad
- andmebaasid
  - TKS test- ja tootebaas
  - Lexi test- ja tootebaas

2.3 TKS süsteemi üleviimise metoodika

Seoses TKS uuendamisega ei ole kinnitatud täpseid TKS 2.0 ja sellele järgnevate versioonide ärivajadusi. Seetõttu võeti eesmärgiks rakenduse loomisel, et see peab võimaldama praeguse andmemudeli edastamist ning olema samaaegselt kujundatav tekkivate ärivajaduste tarbeks. Antud hägusaid ärinõuded arvestades kujunes süsteemi üleviimine järgnevatesse sammudesse, millest osasid tehti samaaegselt:

1. Analüüs olemasoleva süsteemi arhitektuuri ja andmemudeli kohta, kitsaskohtade tuvastamine
2. Veebihaldusliidese loomine
3. Andmete koondamine ning Lexi keskhaldusest edastamisvõimekuse loomine uue süsteemi tarbeks
4. Andmevahetuse loomine TKS 2.0 ja keskhaldusteenuste vahel
5. Uue süsteemi silumine, testimine, diagnostika ja analüüs

Esmase analüüsi ning koosolekute raames täpsustusid süsteemi üldnõuded ning soovid, millega arvestatakse süsteemi loomisel. Ettevõttepoolne soov oli näha uue süsteemi puhul paremat käideldavust ning töökindlust võrreldes vana lahendusega. Selle täpne rakendamine jäeti autori vastutusele lähtuvalt parimatest tarkvaraarendustavadest.

3. Tehnoloogiad

Lõputöö loomisel kasutati erinevaid tehnoloogiaid, mille valimisel tuli kaaluda järgnevaid äri- ning süsteemiarhitektuuri eesmärke:

- Kas antud tehnoloogia...
  - tagab süsteemi jõudluse ning efektiivsuse?
  - limiteerib pärandtarkvara tekke?
  - on äriliselt jätkusuutlik?
- Mis on antud tehnoloogia...
  - konkurentsieelised võrreldes samalaadsete lahendustega?
  - suurimad puudused?

Nendest kriteeriumitest lähtuvalt rajati TKS 2.0 keskne süsteem *.NET platformile*. Kogu projekti andmestik on talletatud SQL Serverisse, mille vahel on loodud ühendus läbi Entity Framework raamistiku. Blazor abil rajatud veebiliideses hallatakse vastavate ühistute ja selle nimistusse kuuluvate poodide ühendusi. Lexi keskhalduskeskkonnast liikuv relevantne info sõnumijärjekorra ja rakendusliideste JSON parsitakse TKS taustsüsteemis lahti ja vastava infotüübi põhjal tehakse täiendavad päringud keskhaldussüsteemi rakendusliidese pihta. Vastava infopäringu baasil luuakse väljundsõnumid mis talletatakse SQL Serveris teatud ajaks kassasüsteemidele pärimiseks. Juhul kui sõnumid aeguvad, ning nende jaoks tekib vajadus, saavad kassasüsteemid tulevikus teha tagasiulatuvaid päringuid, mille puhul pannakse sõnum kokku ühekordseks edastamiseks.

3.1 .NET platform

.NET platform[1] on Microsofti poolt rajatud vaba lähtekoodiga raamistik mis on liidestatud programmeerimiskeelte nagu C# ja F# ning mitmete teekidega. Raamistik vastab ideaalselt ärilistele vajadustele, kuna võimaldab süsteemiarhidektuuris luua jätkusuutliku lahenduse. See omakorda vähendab pärandtarkvara teket kuna arendusmeeskonna kompetents on suunatud tulevikuvaates .NET platformi kasutamisele. Kuigi varasemalt on .NET olnud täielikult seotud Windowsiga ning sellest tingitud suure ressursimahukusega, siis uuematel versioonidel seda piirangut ei ole - see võimaldab vähema resursinõudlusega virtualiseerimist erinevatel platformidel.

Antud raamistiku kasutamine projekti keskmes võimaldab teekide haldamiseks kasutada NuGet paketihaldurit mis võimaldab efektiivselt liidestada projekti täiendavate teekidega. Java paketihaldur Gradle ei võimalda automatiseeritult lisada ega uuendada pakette. Sarnast automatiseeritust pakub Node läbi npm halduri Javascriptile. Npm limitatsiooniks on tema suurima vabavaralise platformi staatus, milles paiknevad paketid on omavahel tihedalt seotud, ning sellega kaasnenud sõltuvus- ja küberintsidentide ajalugu ei anna kindlustunnet süsteemi jätkusuutlikuseks[2][3][4][5]. Platformi limiteerivaks faktoriks on versioonide 3-aasta pikkune toetustsükkel mis on Javaga võrreldes märkimisväärselt lühem. Seetõttu on rakenduse raames kasutuses olev .NET 6.0 uuendamine järgnevale kestsustugi versioonile vajalik juba hiljemalt aastaks 2024.

3.1.1 Entity Framework Core

Entity Framework Core[6] on vabavaraline objekt-relatsioonvastendi ning platformist sõltumatu implementatsioon Entity Frameworkist, mõeldud kasutamiseks .NET platformiga. Selle abil on võimalik luua otseühendus koodis olevate olemite, nendega seostatud andmete, ning neid haldavate andmebaasihaldussüsteemidega. Sarnaselt Hibernate ja teiste ORM tarkvaradega on läbi Entity Frameworki võimalik teha andmebaasipäringuid otse koodi tasandil. EF puhul on lisaks võimalus luua andmebaasimudel otse koodi tasandil kirjutatud olemite põhjal ning automatiseerida andmebaasimigratsiooni kirjed. Sarnast täislahendust on võimalik leida ainult Djangos, mille puhul on see liidestatud ainult Pythoniga. Teiste keelte puhul on sarnane funktsionaalsus jaotatud mitme kolmanda osapoole pakutavate pakettide vahel, näiteks Java puhul Hibernate ja Liquibase, tekitades lisanduvaid lülisid.

Samuti võimaldab EF luua täiendavaid abstraktseid andmestruktuure mille abil on võimalik defineerida asünkroonsed baasoperatsioonid kõikidele olemitele andmebaasi lisamisel, eemaldamisel, uuendamisel ja kustutamisel. Kasutades neid täiendusi on võimalik märkimisväärselt vähendada lähtekoodi täiendamist uute olemite lisamisel.

3.1.2 Blazor

Blazor[7] on WASM (*WebAssembly*) tehnoloogiatele rajatud veebiraamistik, mis võimaldab kirjutada interaktiivseid kasutajaliideseid kasutades JavaScripti asemel C# programmeerimiskeelt. Antud raamistik kasutab C# ja HTMLi süntaksit kombineerivale Razor märgistuskeelele. Kasutajale on võimalik serveerida Blazorile rajatud kasutajaliidest nii serveris kui otse kliendi masinas. Selle abil on võimalik serveerida baitkoodi otse kasutaja brauserisse ning sellevõrra vähendada sõltuvust massiivsetest JavaScript baasil toimivatest raamistikest. Samuti võimaldab see otseühendust kõikide olemite ning teenustega mida üle-

jäänud projekt kasutab. See võimaldab kasutada LDAP rollipõhist autenteerimist teenuse administraatorite määramisel ning vältida lisanduvate rakendusliideste kasutamist.

3.1.3 Serilog

Serilog[8] on logimisliides .NET platformile mis võimaldab luua struktuurseid ning põhjalikke logikirjeid ja neid edastada erinevatesse logimisliidestesse. Serilogiga on võimalik suurendada rakenduse läbipaistvust, luua arusaadavust milline kasutaja andmepäringuid teeb, ning jälgida TKS rakenduse protsesse. Võrreldes teiste logimisteekidega, võimaldab Serilogi logidesse sisestada serlialiseeritud infot selle asemel, et logidesse sisestatakse ToString() olemasolevast objektist. Serilog toetab suurt kogust erinevaid andmeneele mis võimaldab lihtsat integratsiooni ning vajadusel nende väljavahetamist.

3.2 Docker

Docker[9] on operatsioonsüsteemi tasandil toimuv virtualiseerimine mis võimaldab jooksutada kimbustatud tarkvara, teeke, ning nendega seotud konfiguratsioone. Dockeri eeliseks on keerukamate serveritarkvarade lihtne kasutamine tänu selgelt defineeritud ühenduste võimaldades lihtsalt luua arenduskeskkond TKS jaoks. Samuti võimaldab see TKSi vajadusel tulevikus konteineriseerida tänu .NET integratsioonile erinevatel platformidel.

3.3 Elasticsearch & Kibana

Elasticsearch[10] on hajus vabatarkvaraline otsingu- ja analüütikamootor, mis töötab erinevate andmetüüpidega. ELK pinu, mis koosneb Elasticsearchist, Logstashist ja Kibanast, võimaldab rikastatud andmeid koguda ning neid dünaamiiselt anaüüsida. Elasticsearchi eelis on tema ühendatavus Serilogiga mis ühendub otse teenuse Elastic sõlmedega, kaotades Logstashi logide edastamise vajaduse.

Kibana[11] on Elasticsearchi kasutajaliides, kus on võimalik vaadata ning teha analüütikat sisestatud info põhjal. Kibana võimaldab teha reaalaja analüütikat andmetest ja selle põhjal edastada teavet, kui rakenduse käitumises on anomaaliaid.

3.4 SQL Server

SQL Server[12] on Microsofti poolt loodud DBMS (database management system), mis laiendab SQL keele baasstandardit. Andmebaasipakkuja valikul lähtuti selgest eelisest kasutada majasisest andmebaasilahendust mis on varasemalt juba litsenseeritud ning

integreeritud administreerimislahendustega. Kuigi eelis vabavaraliste andmebaaside nagu PostgreSQL või MariaDb puhul seisneb nende platformivabaduses, on nende puhul risk suurem pärandtarkvara tekkel, sarnaselt vana TKS poolt kasutatava Oracle baasi puhul.

3.5 IIS

IIS[13] (Internet Information Services) on Microsofti poolt loodud veebiserveri lahendus mis võimaldab veebimajutamist erinevate veebiteenuste jaoks. IIS valikul lähtuti sarnaselt andmebaasilahenduse valikul tehtud järeldustest. Kuna IIS on juba tugevalt integreeritud teiste keskhaldusteenuste pakkumisel ja omab väga häid koormuse tasakaalustamise- ning integratsioonivõimalusi .NET 6.0 lahendustega, on see kõige jätkusuutlikum veebiserveri lahendus projektile.

3.6 Azure DevOps

Azure DevOps[14] on Microsofti poolt pakutav pilve- või ettevõtterakenduse versioonihaldusplatform, mis baseerub Git versioonihaldussüsteemile. Kuna Azure DevOps on majasiseselt valitud versioonihaldusplatform, siis oli see ainukene tehnoloogia, mille valik oli autorile eelnevalt etteantud. Platformi eeliseks on otsevõimekus integreerida .NET platformi rakendustega ning pakkuda detailsemat pidevvalmidussüsteemi haldamist. Platformile rajatud pidevvalmidussüsteem võimaldab lihtsalt integreerida tarkvara automaatne testimistsükkel, toote- ning teenuskeskondade uuendamine, ning pakettide manageerimine IISi publitseerimisel.

4. Arhitektuur

Lõputöö raames valmis tooteandmete saatmise süsteem, selle toetamiseks vajaminevad andmeliidesed ning andmeühenduse kasutajate haldamise veebiliidesed (Joonis 1).

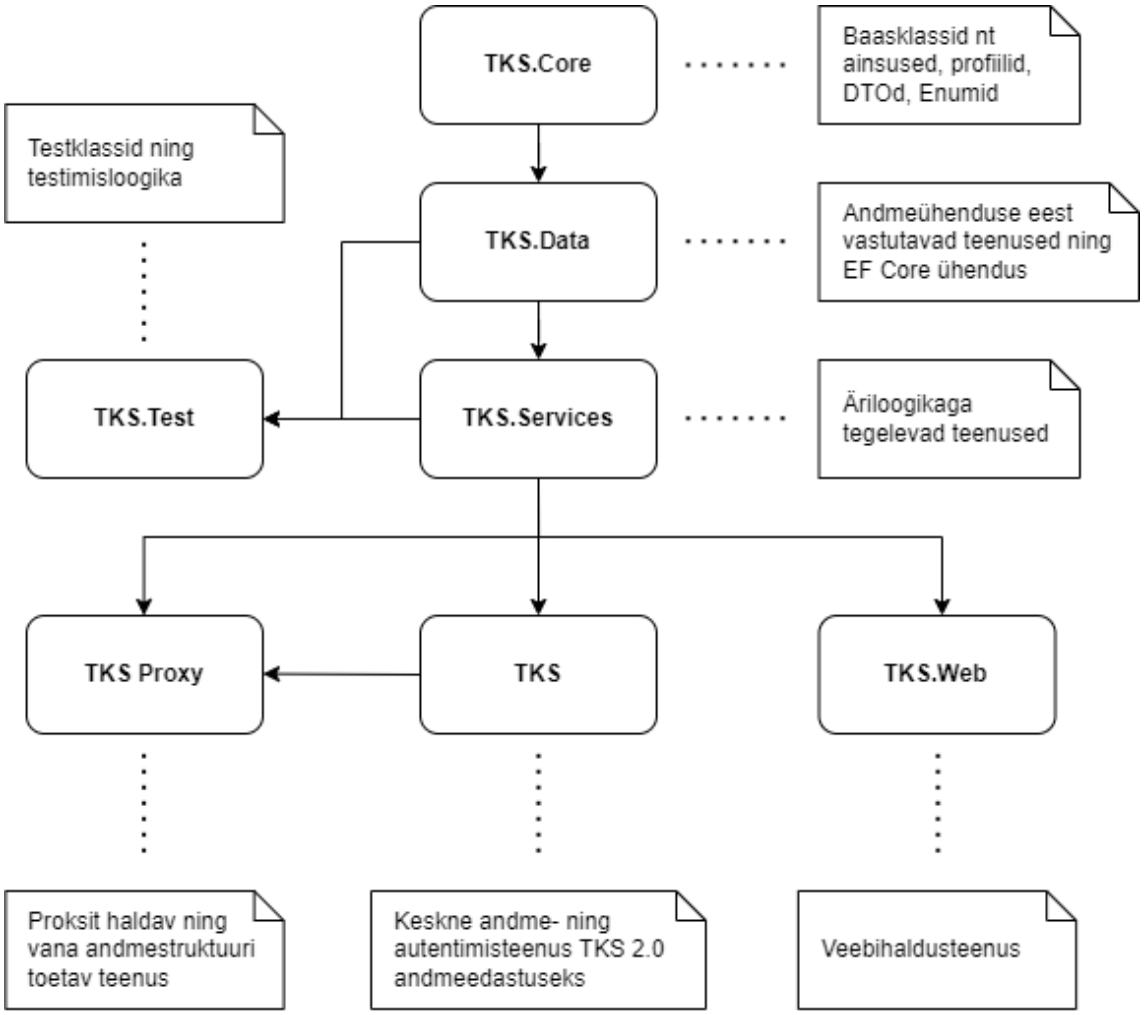
Süsteem on jaotatud kolmeks (3) alamprojektiks:

- TKS - TKS 2.0 keskne süsteem
- TKS.Web - TKS veebihaldusliides
- TKS.Proxy - haldab proksiteenusega seotud süsteemi

Nende 3 alamprojekti toetamiseks on loodud neli keskset üldkasutavat moodulit mis omavahel tagavad teenuseid ning andmeliikumist alamprojektidele. Need neli (4) moodulit on järgnevalt:

- TKS.Core - andmemudelite moodul
- TKS.Data - kontrollib ning seostab andmebaasiühendusi
- TKS.Services - koondab äriprotsesse juhtivad teenused
- TKS.Test - haldab kogu testimisprotsessi

Autor on kõikide, välja arvatud TKS.Proxy, alamprojektide loomisesse panustanud ainuisikuliselt. TKS.Proxy puhul on autor panustamas süsteemi liidestamisse, andmemudelite integreerimisse, ja sujuvasse üleminekusse TKS 1.0 süsteemilt TKS.Proxy vahesüsteemile.

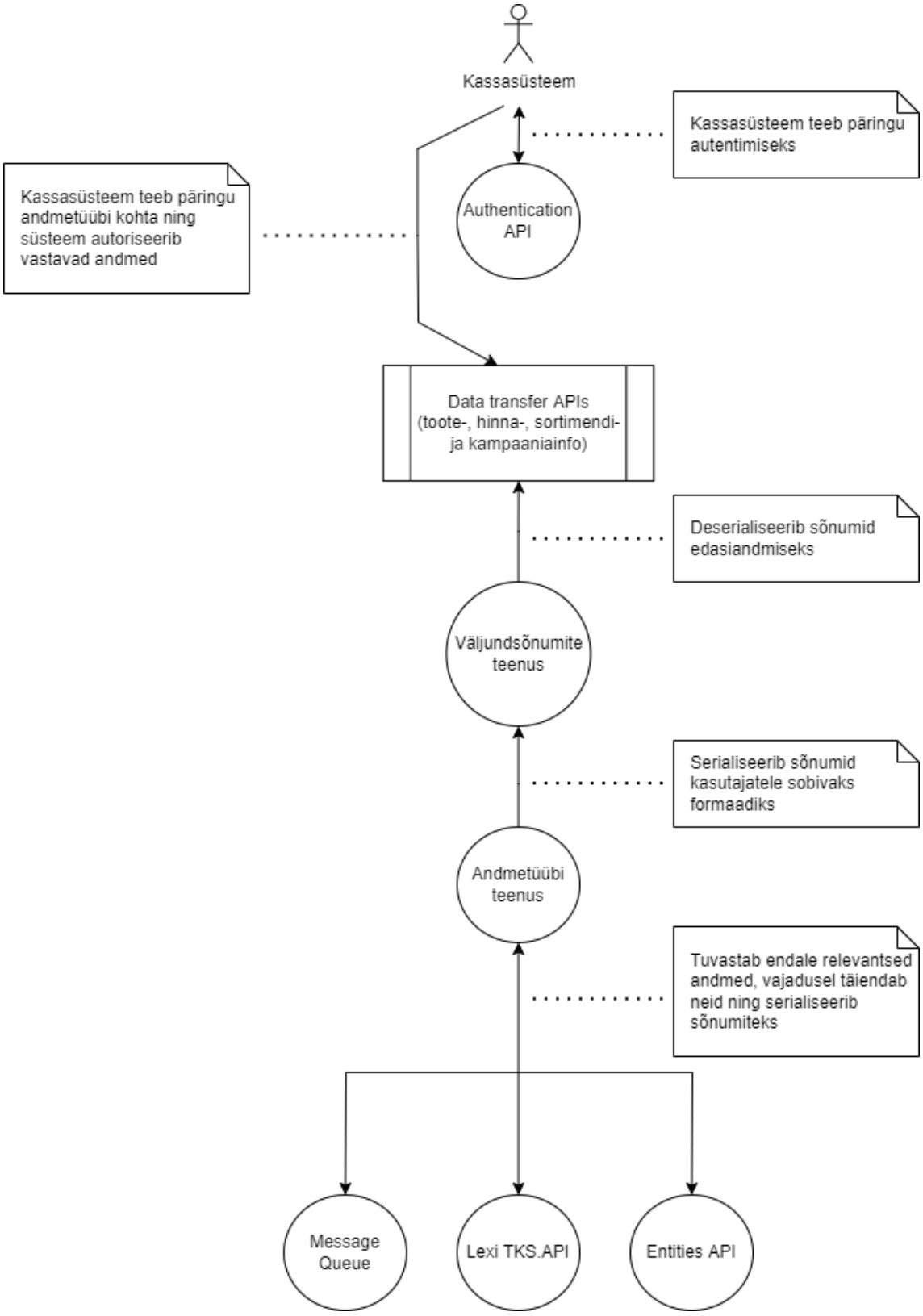


Joonis 1. Süsteemiprojektide sõltuvused

Lexi keskhaldussüsteemi rakendusliideste projekti loodi lisaks uus rakendusliides, mis toetab spetsiifiliselt TKSi toimimiseks vajaminevat funktsionaalsust. Olemasolevatest rakendusliides süsteemidest võeti kasutusele olemiinfo rakendusliides, mille abil on võimalik leida toote- ning kasutajainfot. Kasutusele võeti ka Lexi keskhaldussüsteemis toimuvate regulaarsete muudatuste sõnumijärjekord, mille abil on võimalik teada saada TKS andmemudeli jaoks oluliste andmemuudatuste toimumisest. Olemasolevat sõnumijärjekorra ühenduse automatiseeritud loomist täiendati, vähendamaks selle sõltuvust kolmandate osapoolte teekidest ning lisati TKS 2.0 projekti alglaadimisesse.

TKS süsteem rajati prioriseerides andmevahetuse kiirust ning kasutades võimalikult palju asünkroonset andmevahetust (Joonis 2). Selle otstarbeks rajati Entity Framework Core tasemele hoidlad ja tööühiku abstraktsioon, mis võimaldasid luua paralleelseid ning korratavaid andmepäringuid TKS andmebaasi. Vastavalt sissetulevatele andmetele võtab süsteem kontrolleritasandi info, muudab selle teenusetasandil olemiks, ning talletab peale äriloogika toiminguid vastavasse andmetabelisse. Sõnumijärjekorrast tuleva info puhul

leitakse esimesena ühine tunnus mille abil on võimalik kõigist sõnumitest saada täpne süsteemiinfo kasutades keskhalduse rakendusliideseid.



Joonis 2. Süsteemi üldtöövoog

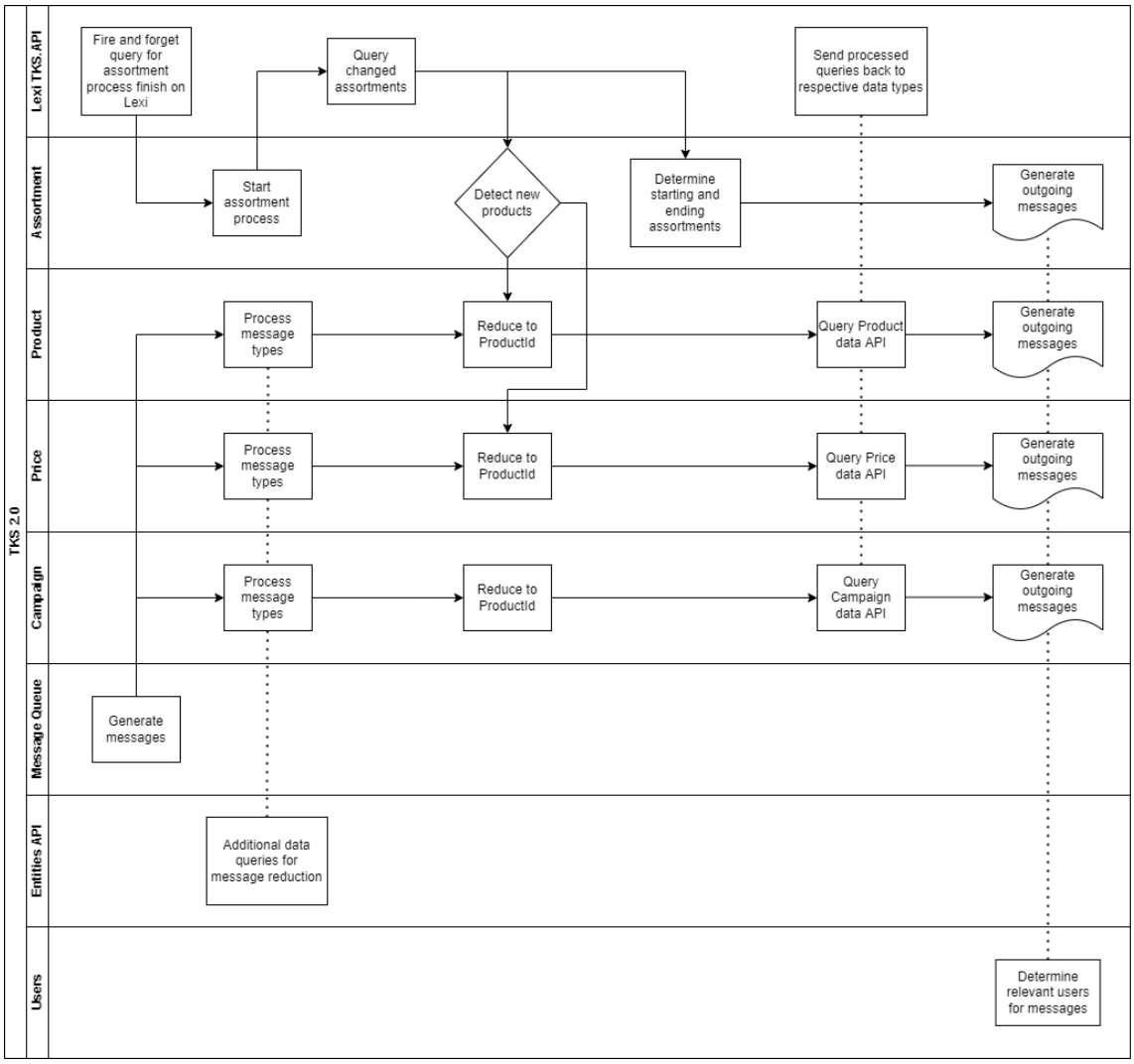
4.1 Sortimendiinfo töövoog

Sortimendiinfo on ainus infoliik mille andmevahetus on ajastatud igapäevaselt, seetõttu on selle infoliigi töövoog märkimisväärselt teistsugune võrreldes teistega. Sortimendiinfo liikumisel keskhaldussüsteemi, ning selle edukal lõppemisel, annab keskhaldussüsteem "saada ja unusta" päringu abil märku, et andmeliikumine on lõppenud. See võimaldab keskhaldussüsteemi ressurssi kokku hoida, ning sortimendihaldussüsteemi vahelise ühenduse lõpetada, ilma seda asjatult lahti hoidmata. TKS reageerib antud päringule ning loob sobiva kuupäevaga seotud info saamiseks üle rakendusliidese ühenduse otse keskhaldussüsteemi.

Keskhaldussüsteem edastab TKSile kahte tüüpi sortimendiinfot: lõppeva ning algava sortimendi kohta. Antud andmete põhjal luuakse väljaminevad sõnumid, mis sisaldavad relevantseid asukohti kassasüsteemidele ja nende asukohtadega seotud sortimenditooteid. Nendele lisatakse ka sortimendihaldussüsteemsit väljajäävad erisused, milleks on kesksortimendis puuduvad asukohapõhised tooted. Juhul kui sortimenti tekivad uued tooted, võimaldab selline kahetasandiline süsteem ka tuvastada uute toodete infot mis ei ole eelnevalt edastatud kassasüsteemi, ning tooteinfo töövoo tõttu ei pruugi saada edastatud. Samuti kontrollib see vajadust uue hinnainfo jaoks kui toode lisatakse kauplusesse kampaaniaperioodiks.

4.1.1 Toote-, hinna- ja kampaaniainfo töövoog

Teiste põhisüsteemide töövoog on rohkem seotud sõnumijärjestuse jälgimisega. Tootega on seotud 19 erinevat sõnumitüüpi, hindadega 3, ning kampaaniaga 1-2. Kuna sõnumite arv mida erinevad infoliigid jälgima peavad on suur, siis on enamus teenused taandatud toote ID leidmisele ning sellekohaste päringute tegemisele läbi Lexi rakendusliideste. See võimaldab teha taaskorratavate andmete päringuid ning lihtsustada sissetuleva info voogu. Toote- ning hinnainfo vajadust analüüsitakse ka sortimendiinfo sisenemisel, kuna uutel sortimenti sisestavatel toodetel on puudu vastav informatsioon kassasüsteemides. Kampaaniainfo süsteem jaotab kampaaniainfo ka kaheks eraldiseisvaks infoliigiks: üks mis koosneb kampaania keskinfost (näiteks kestvus, liik, kaupluste esindus), ning teine kampaanias osalevate toodete infost (näiteks toote soodushinnad, tagastavusinfo, kliendikaardisoodustused) (Joonis 3).



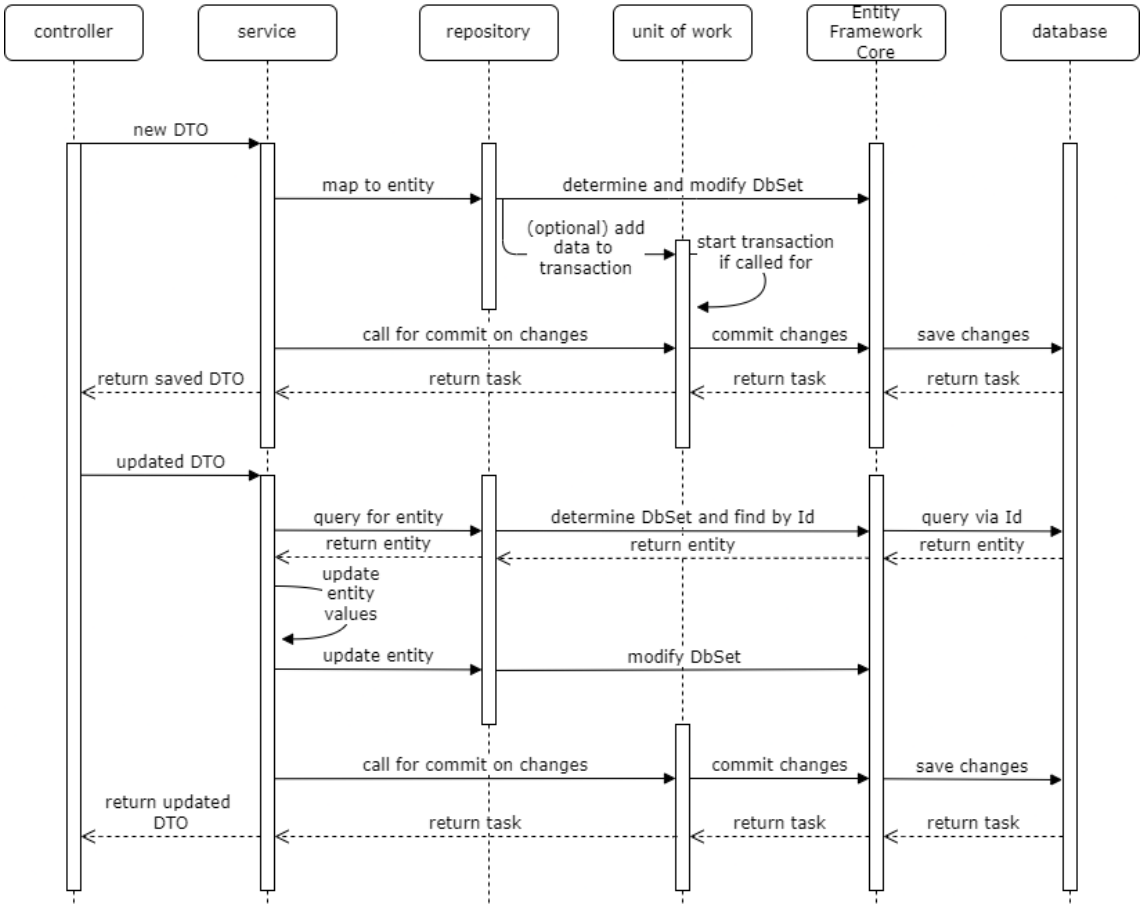
Joonis 3. Protsess väljundsõnumite tekkeks

4.2 Liidestamine andmebaasiga

Entity Framework Core toetab andmebaasimuudatusi läbi klassi DbSet ning andmekonteksti DbContext abil. Need jälgivad muudatuste toimumist ning tegelevad nende rakendamisega baasi tasemel. Kuigi DbContext käsitleb ennast kui Unit Of Work (tööühik) ning Repostitory (hoidla) mustritena, siis neid täiendades on võimalik vähendada koodi duplikeerimist. Täiendada on võimalik transaktsioonide haldamist, mille puhul on tavalisel Entity Frameworki kasutamisel limitatsioonid. Kuigi Entity Framework teeb transaktsiooni põrumisel tagasipöörde, on tekkinud olukordi, kus need tagasipöörded on poolikud, kus osa andmestikust on juba salvestatud baasi ning teine osa pööratakse tagasi, tekitades andmestikku duplikaate. Samuti on DbSet käskluste otsene kasutamine lihtsam anda üle mingile abstraktsioonile, lihtsustades paljude päringute kordust sarnaste operatsioonide kirjeldamisel (Joonis 4).

Operatsioonid korduse vähendamiseks rakendatakse üldiselt .NET projektides hoidla[15] mustrit, kuhu ehitatakse üldine abstraktsiooniklass, mida on võimalik laiendada kõikidele olemitele. Enamasti pakub selline lahendus standardseid andmebaasioperatsioone nagu lisamine, pärimine, muutmine, ning kustutamine; vajadusel ka üldiseid spetsialiseeritud päringuid vastavalt ärivajadustele. See vähendab koodi kordumist ning mugavdab selle haldamist, vähendades märkimisväärselt pärandtarkvara tekke ohtu.

Andmebaasipäringute transaktsioonide halduseks on loodud selle kohale tööühiku[16] abstraktsioon. Selle abil on võimalik ühendada hoidlad üheks koostöötavaks klassiks, võimaldades teenustele konkurentseid päringuid mitme erineva olemi pihta. Samuti saab tööühiku panna vastutama varasemalt välja toodud transaktsioonide jätkusuutlikuse eest, tagades kooditasemel transaktsioonide tagasipöörded. Olemiteenuse tasandil võimaldab see otsest seostatust nii tööühiku, kui ka temaga mitteseotud hoidlate vahel. Samuti võimaldab tööühiku kasutamine andmeühenduse sõltumatust Entity Frameworki pakutud lahendusest jättes võimaluse selle vajaduspõhiseks väljavahetamiseks.



Joonis 4. Andmeühenduse töövoog

4.3 Veebihaldusliides

Keskhalduses olevad andmed on enamasti omavahel seostatud läbi toodete ning asukohtade. TKS andmemudeli edastamiseks on need andmed vaja veel omavahel seostada ühistute ning nende lõpp-punktidega, kuna erinevad ühistud kasutavad kasutavad erinevaid kassasüsteeme - osad isegi mitut. Kuna antud seosed ei ole omavahel otseselt loodud keskhaldussüsteemis, siis on selle seose haldamiseks loodud vanas süsteemis veebihaldusliides, mille ületoomine uue süsteemi jaoks ei olnud otstarbekas. Süsteem on vana ning selle liidestamiseks uue TKSiga oleks vaja teha lisanduvaid APIsid. Selle asemel on mõistlik luua uus TKS veebiteenus, mis suhtleks otse TKS andmebaasiga.

Blazori põhjal rajati veebiteenus, mis võimaldas omavahel luua ja siduda otspunkte erinevate ühistute ning nende kaupluste vahel (Lisa 2). Uue süsteemi eesmärgiks on vähendada üleliigse info hoidmist - seetõttu ühendat veebiliides olemiliidesega Lexi keskhaldussüsteemist, vältimaks otsest andmebaasiühendust keskhaldussüsteemiga. Kaupluste seostamiseks ühistutega loodi kasutajahaldusteenus *UserManagementService*. Teenus pärib olemiliideselt ühistute kohta aktuaalset infot ning võimaldab sinna liita ühistule seotud kauplusi, vähendamaks omavahelist segunemist erinevate ühistute ning nendega mitte seotud kaupluste vahel. Samuti võimaldab süsteem mitut eri tüüpi kassaserverit kasutavaid ühistuid hallata, jaotades asukohad erinevate tarbijate vahel.

4.4 Logimine ja monitooring

Tänase TKSi suurimaks limitatsiooniks on ajaloo- ja logimisandmete puudumine. Süsteemi edukas toimimine on teada ainult selleks hetkeks kui andmete olemasolu muutub oluliseks poe süsteemide jaoks. Käsitsi veatuvastus ei ole jätkusuutlik uue süsteemi lahendusel. Selle otstarbeks võeti kasutusele Serilog, mis võimaldas otsest andmeühendust erinevatesse andmekaussidesse. Samuti võimaldab see kirjutada struktuurseid logisid ning neid analüüsida.

Serilogi konfiguratsiooni põhjal leidusid kõige otstarbekamad lahendused logida andmeid otse Elasticsearchi, ning samuti hoida tagavaruks lokaalseid logifaile TKS serveris. Antud lahenduste põhjal on Kibanas loodud töölaud, mis võimaldab teha reaalajas analüüsi. Samuti on logifailide põhjal üles seadud Zabbix monitooring, mis teavitab teenusehaldureid riketest ning ebakorrapärasest toimingutest.

4.5 Autentimine ja autoriseerimine

Kuna TKSi ning kassasüsteemide vaheline liiklus toimub üle avaliku võrgu, on kriitiline, et TKS teenused oleks krüpteeritud ning kasutatavad ainult autenditud ning autoriseeritud kasutajatele. TKS siseliiklus on krüpteeritud IIS serveris ja välisveebi ühenduse krüpteeringu eest vastutavad Coop süsteemiadministraatorid. Autentimise ning autoriseerimise saavutamiseks oli kõige mõistlikum lahendus kasutada Bearer token autentiteerimist mis võimaldas tuvastada erinevaid kasutajaid ning edastada neile relevantseid andmeid päringute käigus. TKSi kasutava kassasüsteemi tööprotsessi puhul on eelnevalt vaja pöörduda autentimisteenuse poole, mis tuvastab kassasüsteemi edastatud andmete põhjal tema identiteedi, ning väljastab selle alusel *JWTi* (JSON Web Tokeni). Selle tokeni alusel on võimalik kasutaval kassasüsteemil pärida endale vajaminevaid andmeid vastavatest kontrolleritest. Kontrolleril on selle abil lihtne aru saada ning serveerida vastavalt temale sobivat informatsiooni.

4.6 Teenuse levitamine

Azure DevOps pakub karbist-välja-liidestust paljude .NET ning seotud platformide ühendamiseks. Peale versioonihalduse oli kriitiline, et TKS 2.0 oleks kiiresti levitatav nii test- kui toodangukeskondadesse, tagades kõrge aktiivaja ning tagasipöörde võimaluse vigade esinemisel. Selle otstarbeks loodi konveier, mis vastutab mõlema keskkonna automaathalduse üle ehitades rakenduse, hallates automaatselt vajaminevaid NuGet pakette ning testides aplikatsiooni vastavalt ainsus- ning integratsioonitestidega. Samuti võimaldab Azure otsest levitamist IIS platformile ning konfiguratsioonide automaatset edastamist.

Kuna IIS on juba eelnevalt laialdaselt kasutuses majasisestes projektides, vajas see vaid uuendust ning konfigureerimist .NET 6.0 standardile. Kõik kolm TKS teenust on hallatud eraldiseisvate veebiteenustena, mida uuendatakse vastavalt Azure DevOpsi abil. Läbi IIS sisseehitatud funktsionaalsuse, on kõigi kolme resursivajadused ning aktiivsus automaatselt hallatud, tagades serveri resursside säästimise madala kasutusega perioodidel.

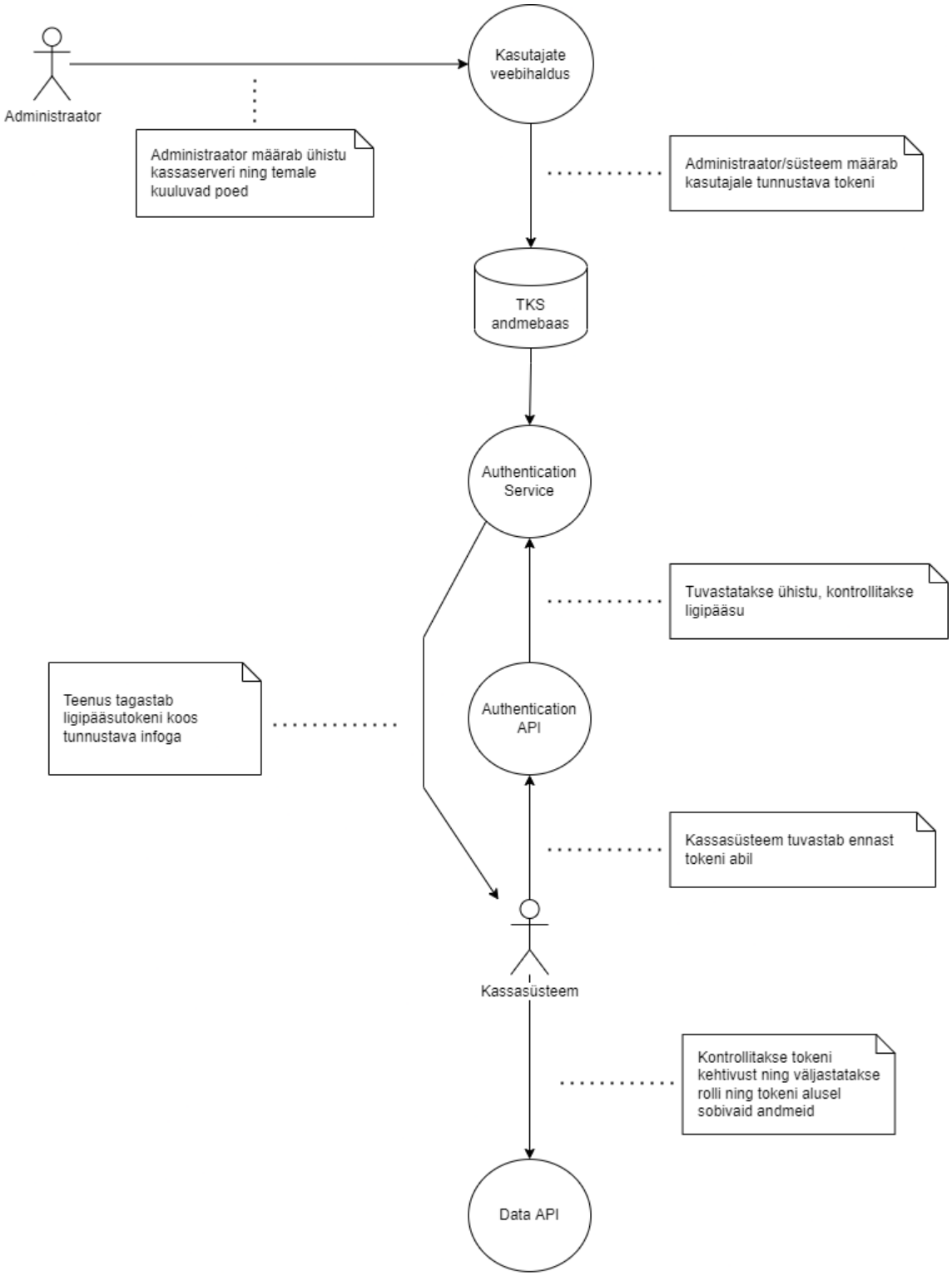
5. Kasutamine

5.1 Veebiliidese kasutamine

Veebiliides on esimene samm TKS 2.0 kasutamisel. Selle põhjal saab määrata süsteemi kasutajaid, luua neile ligipääsuandmed, ning seostada nende jaoks vajaminevad kauplused. Samuti saab TKS proksi siit infot seoses erinevate kassasüsteemide rakendusliideste kohta. Selleks, et tagada keskkonna turvalisus, on keskkond autenditud LDAP abil ning sisselogimiseks on vaja olla andmehalduri rollis. Samuti on keskkond kättesaadav ainult siseteenusena, kaitstes selle kättesaadavust väliste rünnete vastu.

5.2 TKS 2.0 kasutamine

TKS 2.0 puhul on süsteem rakendusliideste abil toimiv veebiteenus, mille külge saavad ennast ühendada erinevad kassasüsteemid. Selle kasutamiseks on vaja kassasüsteemidel ennast eelnevalt kinnitatud ligipääsuandmetega autentida ning selle alusel väljastatakse neile Bearer token, mille abil süsteem autoriseerib kassasüsteemile sobiva info saamiseks (Joonis 5).



Joonis 5. Autentimise töövoog

Kassasüsteemil on võimalus pärida ühte neljast erinevast andmetüübist:

- sortimendiandmed;
- hinnaandmed;

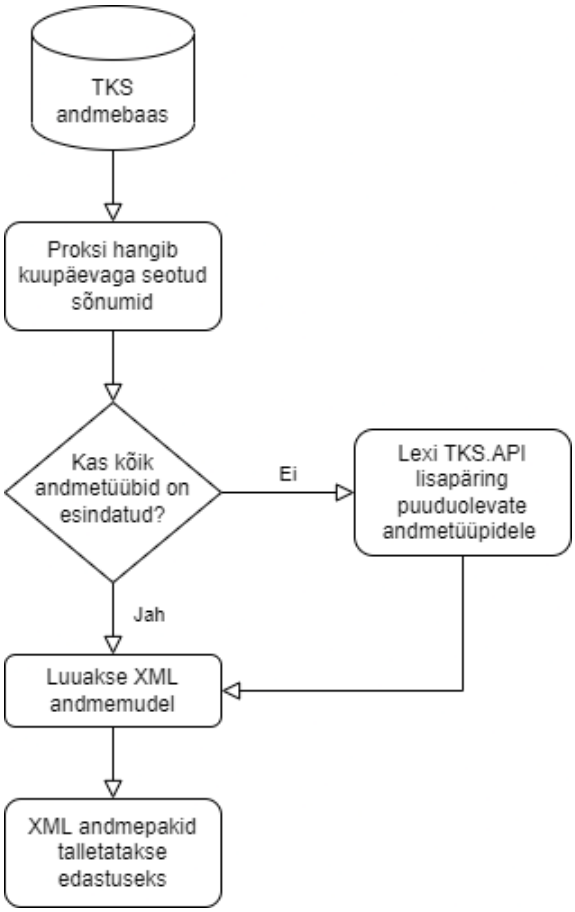
- tooteandmed;
- kampaaniandmed, mis jagunevad omakorda:
  - kampaaniaga seotud andmeteks;
  - kampaanias osalevate toodetega seotud andmeteks;

Kõigi nelja andmetüübi jaoks on rajatud eraldi JSON andmemudelit edastavad kontrollerid. Kontrolleriga ühendusel tuleb kassasüsteemil esitada oma JSON veebitoken, mille analüüsil selgitatakse vastava kassasüsteemi sõnumid, ning need edastatakse. Kuna andmeesitus ei pruugi alati olla edukas, hoitakse andmeid teatud ajani (hetkel seadistatud 14 päeva) süsteemis, et neid saaks kiiresti edastada juhul kui info jääb puudu või andmevahetus toimub veidi hiljem kui soovitakse. Väljaspool seda perioodi on võimalik pärida kõiki andmeid ning neid luuakse vastavalt vajadusele, kuid antud toiming võtab kauem andmete taasloomise tõttu. See võimaldab uue poe tekkel edastada kõik relevantsed andmed ilma, et peaks tekitama käsitsi loodud andmeülekandmislahendusi.

5.3 Proksi kasutamine

TKS 2.0 proksi on ühendatud otse TKS andmebaasiga, tagades võimaluse tarbida samu andmeid mida edastaks TKS 2.0, ilma, et oleks vaja autenteerida ning pärida neid üle rakendusliidese. Kuna TKS 2.0 andmemudel on palju granulaarsem, siis ühe muudatuse edastamiseks vajab vana andmestruktuur kõiki andmeid kõikidest andmetüüpidest. Seetõttu on proksi ühendatud ka Lexi keskhalduses paiknevate rakendusliidestega, et täiendada puuduolevat andmetüübi informatsiooni.

Selle hõlbsustamiseks hoitakse kõikide TKS 2.0 sõnumite puhul infot millise tootega antud toiming toimub. Keskselt on need andmed seotud toodetega, ning sellest lähtuvalt hoitakse iga sõnumiga alles toote ID seost. Protsessi käivitumisel tuvastab proksi kõik kuupäevaga seotud sõnumid, selgitab välja olemasolevad andmetüübid, ning täiendab seda puuduolevate andmetega Lexi rakendusliideste abil. Seejärel loob proksi vana süsteemi andmemudeli põhjal XML andmepaki kõigist neljast andmetüübist ning talletab selle6. Kättesaamiseks teeb kassasüsteem SOAP teenuse päringu vastu proksi otspunkti ning proksi edastab eelnevalt genereeritud XML andmepaki. Andmepakk edastatakse kord päevas ning on tagantjärgi päritav selle hoiuperioodi lõpuni, mille järel see kustutatakse.



Joonis 6. Proksi andmepakkide loomine

6. Tulemuste analüüs

Lõputöö eesmärk oli luua uus lahendus, mis vastaks süsteemiarenduse töökindluse, käideldavuse, hoolitavuse printriipidele võrreldes vana lahendusega. Antud peatükis analüüsitakse ning võrreldakse vana ja loodud süsteemi vastavust nendele kolmele printsiibile. Esineb peamiselt kolme tüüpi kasutajaid kelle rollideks on:

- arendaja
- kassasüsteem
- andmehaldur

6.1 Töökindlus

Süsteemi töökindlust hinnates peab eraldiseisvalt vaatama tarvara disainimisel haldatavaid printsiipe ning inimfaktoreid, mis mõjutavad süsteemi kasutamist. TKS automatiseeritud andmevahetuse tõttu on tugevalt rõhk süsteemi disainil ning selle hindamiseks ja määratlemiseks saab hinnata RAM (reliability, availability, maintainability) mudeli järgi.

6.1.1 Liiasus

Vana TKSi iseloomulikuks osaks oli andmete haldamine ning vajadusel nende parandamine, mis tähendas arendaja jaoks nende käsitsi haldamist. Ükskõik millise vea esinemisel ei ole võimalik tuvastada ega korrigeerida andmeid vastavalt enne kui lõppkasutaja või kassasüsteemide haldur vea tekkest teada ei ole andnud. Toimingud selle jaoks on ajakulukad ning andmevahetuse automaatse toimumise tõttu on vaja teada täpselt mis andmed vajavad muutmist ning uuesti loomist kassasüsteemide jaoks, tekitades riski inimvigade tekkeks. Kogu liiasus on vana süsteemi puhul ainult arendaja poolt käsitsi hallatav.

Uue süsteemi puhul on kogu tarkvarakasutuse tsükkel liigne ning automaatne. Pidevvalmidussüsteem võimaldab automaatselt testida ning publitseerida uusi arendusi teenuse test- ja tarbekeskkonda ning neid vajadusel tagasipöörata. Andmepäring on muudetud kassasüsteemide jaoks korratavaks, võimaldades kassapoolsete vigade esinemisel kassasüsteemidel ise pärida andmeid ilma, et arendaja peaks vahele astuma ning käsitsi uuesti päringu looma.

Näiteks on tänasel süsteemil selgelt väljakujunenud minimaalselt 20-minutine andmebaasivaadete loomise aeg, millele lisandub vajadusel teiste andmebaasitoimingute tegemine. Seetõttu on isegi puuduvate muudatuste puhul süsteem töös. Samal ajal on uue süsteemi töö seotud andmevahetuse kogusest, mis on suuresti varieeruv ning toimub rohkem kui kord päevas. Liites omavahel andmevahetuse ning sõnumite loomise aja uues süsteemis, saab võrrelda seda vana süsteemi ajakuluga (Tabel 1).

Tabel 1. Ajakulu vana süsteemi ja TKS 2.0 süsteemi andmete loomisel.

Kuupäev	**Ajakulu vanas süsteemis (minutit)**	**Ajakulu uues süsteemis (minutit)**	**Keskmine ajavõit (minutit)**
27.03.2022	50	10	40
28.03.2022	53	14	39
29.03.2022	51	17	34
30.03.2022	49	14	35
31.03.2022	30	9	21
01.04.2022	24	8	16
02.04.2022	22	3	19
03.04.2022	24	11	13
04.04.2022	27	9	18
05.04.2022	24	6	18
Kokku	354	97	253

6.1.2 Testimine

TKS 1.0 süsteemi sidestus oma andmebaasiga võimaldas süsteemi ainult manuaalselt testida. Uue lahenduse puhul on arvestatud sellega, et süsteemi kesksed komponendid on vaja katta üksustestidega, mida on võimalik kasutada pidevvalmidussüsteemi torus automaatselt. Samuti on lisatud teatud äriprotsesside toimimise kontrolliks integratsioonitestid, mis jälgivad andmete liikumist keskhaldussüsteemide ja kassasüsteemide vahel. See tagab uute lahenduste komplekteerivust olemasolevate struktuuridega, vähendades riski võrreldes käsitsi loodud uute andmebaasipäringutega.

Süsteemi andmebaasiliides on kaetud 55 üksustestiga, tagades lisanduvate olemite puhul meetodite korrektset kasutamist. Kaetud on tööühiku puhul transaktsioonide haldamine ning erinevate transaktsioonide staatused. Samuti on kaetud nii olemitega tegelevad teenused kui ka nende liidesed. Integratsioonitestide puhul on jälgitud kogu andmeliiku-

mine alates keskhaldussüsteemidest liikuvast infost kuni kasutajatele loodud sõnumioperatsioonideni.

6.2 Käideldavus

Süsteemi käideldavus on paranenud märkimisväärselt võrreldes vana lahendusega. Andmete liikumine ei ole enam limiteeritud korra päevas ning arvestab andmete liikumise kiirusega. Teenus on limiteeritud ainult sortimendiinfo korrapärase liikumise läbi ning sellelgi juhul saab teenus alustada kohe peale andmete liikumist, vältimaks *cron* tööd.

Süsteemi kõrge aktiivaeg on oluline kassasüsteemide andmevahetuseks, sest selle põhjal on võimalik äriprotsesse hakata ümber muutma. Kui praeguse protsessi vaates toimub andmevahetus kord päevas, siis TKS 2.0 modulaarne, ning osaliselt asünkroonne disain, võimaldab muuta andmete vahetust granulaarsemaks. Parimal juhul, võrreldes vana süsteemiga, on võimalik sõnumijärjekorra kaudu vahetatavaid andmeid kauplustele edastada peaaegu ööpäev varem. Samuti on projekti tihedalt integreeritud pidevvalmidussüsteem ning logimine on teinud käideldavuse hindamise võimalikuks. Nende meedete abil on võimalik vältida süsteemisiseseid tõrkeid, mille olemasolust puudub info teenuse administraatoritel ning arendajatel.

6.2.1 Modulaarsus

Vana süsteemi puhul oli tegemist monoliitarhitektuuriga, mille muutmist ning asendamist on vaja teha kogu tarkvara ulatuses. Selle tõttu on ka andmemudel üks ning selle muutmiseks on vaja viia sisse muudatused kõikide kassasüsteemide ulatuses. Uue lahenduse puhul on andmemudel liidestatud eraldi tükkideks; uue süsteemi vaates neljaks tükiks, mida on võimalik juurde luua, muuta ning jooksvalt täiendada. Selle otstarbeks on andmevahetus keskhaldussüsteemi ja TKS vahel muutunud keerukamaks, kuid modulaarsemaks. Vähendatud sõltuvus 1:1 andmebaasikopeerimisest läbi sõnumiahela võimaldab tulevikus erinevate komponentide väljavahetamist kogu TKS 2.0 teenuse ulatuses.

6.3 Hooldatavus

Vana süsteemi hooldatavust pärssis süsteemi monoliitsus, kompleksus ning vähene halduskompetents oma vanuse tõttu. Kuigi uue süsteemi arendusega on alati oht, et ka selle elutsükli jooksul muutub see pärandtarkvaraks, siis on TKS 2.0 loomisel tugevalt panustatud selle limiteerimiseks. Enamus teenust toetavate liideste ning pakkide puhul on lähtutud asendamise võimekusest, tagades süsteemi osade jätkusuutlikuse. TKS 2.0

puhul on kriitiline .NET platformi suuremate versioonimuudatustega kaasaskäimine, kuna kestsustugi selle jaoks on circa 3 aastat. Vastasel juhul ohustab läbi välisveebi kättesaadava teenuse kaudu potensiaalsed turvanõrkused ning nendest tulenevad küberrünnakud.

6.3.1 Diagnostika

Teenusepoolsete vigade jälgimiseks on seadistatud struktuurne logimine ning nende logide automatiseeritud jälgimine, mis teavitab koheselt süsteemivigade tekkest. See lihtsutstas vigade reprodutseerimist arendajate poolt ning kiirendab nende parandamist märkimisväärselt. Samuti on teenuse elutsükliinfo põhjal võimalik teha ärianalüüsi ning leida kitsaskohti.

7. Kokkuvõte

Töö peamine eesmärk oli luua uus rakendus tootesüsteemile mis oleks jätkusuutlik, kõrge käideldavusega ning läbipaistev. Äriliste protsesside toetamiseks on lisanduvalt omavahel vaja integreerida uue süsteemi andmemudel vana andmestruktuuri mudeliga, ning toetada sellekohast proksit kuni selle ajakohase väljavahetamiseni. Teine eesmärk oli luua veebihaldusliides, mis võimaldaks ühistute ning kassasüsteemide vahelisi ühendusi hallata.

Peamine eesmärk täitus luues .NET raamistikul töötavad veebiteenused mis tarbivad, haldavad ning edastavad kassassüsteemidele vajaminevat informatsiooni. Süsteemiarhitekduuri haldamisel loodud teenus on kõrgelt modulaarne, läbipaistev ning tulevikukindel vastamaks ärivajadustele. Teenus on lahtiliidestatud monoliitteenusest ning selle loomisel jälgiti kõiki parimaid tavasid, et vältida pärandtarkvara teket. Teenuse sujuvaks tööks lisati projektile pidevvalmidussüsteem ning täiendati rakendusega seotud andmete analüütika- ja logimisplatvorme. Lisaks panustas autor proksiteenusesse mis tarbib põhiteenuse loodud informatsiooni ning vajadusel täiendab seda Lexi keskhalduses loodud rakendusliideste abil. Sellise lahenduse abil on võimalik luua vana süsteemiga sarnane andmepakett, mis tagab kuni uue süsteemi kasutamiseni kassasüsteemide andmevajaduse.

Teise eesmärgi täitmiseks loodi Blazor serveripoolne veebirakendus. Rakendus tarbib Lexi keskhaldussüsteemi infot ning loob teenuseadministaatori sisendi põhjal andmeseosed kassasüsteemide ja kaupluste vahel. Lisaks täiendati süsteemi turvalisust, lisades süsteemile läbi .NET identiteediliidese LDAP autentimine ja autoriseerimine.

Kasutatud kirjandus

[1] *.NET*. [Accessed: 31-03-2022]. URL: https://dotnet.microsoft.com/en-us/.

[2] *OWASP NPM Security*. [Accessed: 05-05-2022]. URL: https://cheatsheetseries.owasp.org/cheatsheets/NPM_Security_Cheat_Sheet.html#5-audit-for-vulnerabilities-in-open-source-dependencies.

[3] *Package Planting: Are You [Unknowingly] Maintaining Poisoned Packages?* [Accessed: 05-05-2022]. URL: https://blog.aquasec.com/npm-package-planting.

[4] *kik, left-pad, and npm*. [Accessed: 05-05-2022]. URL: https://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm.

[5] *Dev corrupts NPM libs 'colors' and 'faker' breaking thousands of apps*. [Accessed: 05-05-2022]. URL: https://www.bleepingcomputer.com/news/security/dev-corrupts-npm-libs-colors-and-faker-breaking-thousands-of-apps/.

[6] *Entity Framework Core*. [Accessed: 31-03-2022]. URL: https://docs.microsoft.com/en-us/ef/core/.

[7] *Blazor*. [Accessed: 31-03-2022]. URL: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor.

[8] *Serilog*. [Accessed: 31-03-2022]. URL: https://serilog.net/.

[9] *Docker*. [Accessed: 20-04-2022]. URL: https://www.docker.com/.

[10] *Elasticsearch*. [Accessed: 31-03-2022]. URL: https://www.elastic.co/.

[11] *Kibana*. [Accessed: 31-03-2022]. URL: https://www.elastic.co/kibana/.

[12] *SQL Server*. [Accessed: 18-04-2022]. URL: https://www.microsoft.com/en-us/sql-server/sql-server-2019.

[13] *IIS*. [Accessed: 15-04-2022]. URL: https://www.iis.net/.

[14] *Azure DevOps*. [Accessed: 15-04-2022]. URL: https://azure.microsoft.com/en-us/services/devops/.

[15] Martin Fowler et al. “Repository”. In: *Patterns of Enterprise Application Architecture*. 2003, pp. 322–325.

[16] Martin Fowler et al. “Unit of Work”. In: *Patterns of Enterprise Application Architecture*. 2003, pp. 184–190.

Lisad

Lisa 1 - Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Magnus Muru

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Tooted Kassasüsteemi infosüsteemi versioon 2.0 arendus", mille juhendaja on Gert Kanter
   (a) reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   (b) üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadust ning muudest õigusaktidest tulenevaid õigusi.

30.05.2022

Lisa 2 - Administraatori vaade kasutajahalduse veebiliideses