TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Sebastian Sõeruer 190506IADB

Tarnijate tooteinfo haldamise rakendus veebipoele Büroomaailma näitel

Bakalaureusetöö

Juhendaja: Kristiina Hakk
PhD

Tallinn 2023

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: Sebastian Sõeruer

24.04.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua tooteinfo haldamise lahendus ettevõtte Infotark AS veebipoele. Luuakse rakendus, mis võimaldab tarnijate tooteinfot alla laadida, seda filtreerida, teisendada ja veebipoele edastada.

Lõputöö kirjeldab probleemi lahendamiseks loodud tarkvara arenduskäiku. Kõigepealt seletatakse lahti probleemi olemus, võrreldakse erinevaid olemasolevaid lahendusi, valitakse välja sobivad lahendused rakenduse loomiseks, kirjeldatakse lahenduse arenduskäiku ja analüüsitakse antud tarkvara kasutegurit ettevõttes.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 41 leheküljel, 6 peatükki, 16 joonist, 9 tabelit.

Abstract

Webshop Application for Automatic Management of Vendors Product Info on the Example of Büroomaailm

The aim of this bachelor's thesis is to create a product information management solution for the online store of Infotark AS. An application will be developed that allows suppliers to download product information, filter it, convert it, and transmit it to the online store.

The thesis describes the software development process for solving the problem. First, the nature of the problem is explained, different existing solutions are compared, suitable solutions are selected for creating the application, the development process of the solution is described, and the usefulness of the software for the company is analyzed.

The thesis is in Estonian and contains 41 pages of text, 6 chapters, 16 figures, 9 tables.

Lühendite ja mõistete sõnastik

.NET	Tarkvaraarenduse raamistik
Agiilne arendus	*Agile development*, arendusmeetod, kus kasutatakse lühikesi korduvaid arendustsükleid, kus iga kordusega on võimalik vajadusel kohandada töö skoopi
BLL	*Business Logic Layer*, – äriloogikakiht
CSV	*Comma-Separated Values*, andmefaili formaat
DAL	*Data Access Layer*, andmetele juurdepääsu kiht
DTO	*Data Transfer Object*, ehk andmeedastusobjekt
EAN	*European Article number*, ehk standardne toodete nummerdussüsteem
Excel	Microsoft Excel, tabelarvutus- ja tabeltöötlusprogramm.
FTP	*File Transfer Protocol*, standardne võrguprotokoll, mida kasutatakse failide edastamiseks
Git	Versioonihalduse tarkvara
HTTP	*Hypertext Transfer Protocol*, võrguprotokoll, mida kasutatakse veebiresursside edastamiseks internetis
IDE	*Integrated Development Environment*, integreeritud arenduskekkond, koodiredaktor
JavaScript	Objektorienteeritud programmeerimiskeel, peamiselt kasutatud veebiarenduseks
JSON	*JavaScript Object Notation*, – JavaScriptil põhinev andmevahetus vorming
PHP	*Hypertext Preprocessor*, serveris kasutatav interpreteeritav programmeerimiskeel
SQL	*Structured Query Language*, struktureeritud päringukeel
SSH	*Secure Shell Protocol*, turvalise ühenduse protokoll
XML	*eXtensible Markup Language*, laiendatav märgistuskeel
XSLX	*Excel Spreadsheet XML*, Microsoft Exceli failivorming

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

E-kaubanduse populaarsus on väga kiires kasvus, Eestis kasvab e-kaubandus keskmiselt 25-27% aastas, pandeemia ajal oli kasv aga koguni 50% [1]. Hinnanguliselt ostab veebipoodide kaudu kaupa 76% elanikkonnast [1]. Seetõttu on otstarbekas laiendada oma tootevalikut just eelkõige e-kanaleid silmaspidades.

Ettevõte soovib laiendada müüdavate kaupade sortimenti selliselt, et neid ei peaks ostma sisse enda lattu, vaid saaks tarnijatelt huvipakkuva tootevaliku liidestada veebipoega ja kaup osta lattu alles siis, kui klient on selle veebipoest tellinud.

Käesoleva töö eesmärk on luua rakendus, mis võimaldab tarnijate tooteinfo allalaadimist, selle filtreerimist ja teisendamist ning veebipoodi edastamist. Lahendus peab olema võimalikult automatiseeritud ja töötama minimaalse kasutaja sekkumisega.

Töö esimeses osas antakse ülevaade praegusest seisust ja püstitatakse ülesanne. Teises osas analüüsitakse loodava lahenduse nõudeid, vaadatakse üle võimalikud olemasolevad lahendused ning valitakse ülesande lahendamiseks kõige sobilikumad vahendid. Kolmandas osas antakse ülevaade rakenduse arendusetappidest ja projekti struktuurist. Viimases osas kirjeldatakse valmis rakendust, selle struktuuri, antakse ülevaade suurimatest väljakutsetest ja tuuakse välja võimalikud edasiarendused.

2 Ülesande püstitus ja projekti eesmärk

Käesolevas peatükis tutvustab autor ettevõtte tausta, autori rolli ettevõttes ja olemasolevat protsessi tooteinfo edastamiseks veebipoodi. Lisaks sõnastatakse töö eesmärk.

2.1 Taust

Ettevõtte Infotark AS (Büroomaailm) on Eesti suurim bürootarvete, -tehnika ja -mööbli müügifirma ning ettevõttel on kokku kaheksa kauplust üle Eesti. Ettevõte tegeleb nii äriklientidele kui ka jaeklientidele müügiga. [2]

Töö autor töötab ettevõttes IT-juhi ametikohal, kus igapäevaste ülesannete hulka kuulub probleemide kaardistus, nendele lahenduste leidmine ja olenevalt vajadusest ning kompetentsist ise ülesannete lahendamine või sobiva arenduspartneri leidmine.

Büroomaailma tootevaliku haldus käib hetkel manuaalselt, ehk tootevaliku laiendamiseks tuleb igale tootele luua käsitsi tootekaart ettevõtte ERP süsteemi, lisada kaup sortimenti ja osta kaup lattu. Kitsa tootevalikuga on antud protseduur hallatav, kuid tahtes pakkuda klientidele laia kaubavalikut, on käsitsi tooteid luua väga ebaefektiivne. Lisaks on vajalik osta kaup lattu sisse, see ladustada, mis toob endaga kaasa rahalise kulu ja laopinna vajaduse. Kaupluste kaubavaliku laiendamine on ka keeruline, kuna seal seab piirid ette kaupluse enda ruum kuhu kaup välja panna.

Ettevõttes on kasutusel majandustarkvara Microsoft Dynamics AX, mis on liidestatud veebipoega, mis kasutab Magento 2 tarkvara. Tooted luuakse majandustarkvarasse, märgitakse neile külge veebimüügi tunnus ja seejärel saadab automaatliides andmed veebipoodi, kus tooted muutuvad klientidele ostetavaks. Sel viisil saab klientidele pakkuda ainult piiratud tootevalikut, tarnijatel on toodete valik oluliselt suurem. Antud kitsaskoha lahendamiseks oleks vajalik pakkuda klientidele suurt tootevalikut nii, et neid ei peaks varasemalt sisse ostma, vaid saaks seda teha alles peale seda, kui klient on toote ostnud.

2.2 Ülesande püstitus

Tänasel päeval ei ole mõistlik osta kõiki müügiks mõeldud kaupu lattu seisma vaid tänu kiiretele tarnemeetoditele on võimalik tarnida ostetud kaup mõistliku aja jooksul otse tarnijalt.

Ettevõte soovib laiendada müüdavate kaupade sortimenti selliselt, et neid ei peaks ostma sisse enda lattu, vaid saaks tarnijatelt huvipakkuva tootevaliku liidestada veebipoega ja kaup osta lattu alles siis, kui klient on selle veebipoest tellinud.

Väga suurel osal tarnijatel, kellelt ettevõte ostab kaupu sisse on olemas masinloetaval kujul väljundid pakutava tootevaliku kohta. Ettevõtte sooviks on neid tooteid valikuliselt kuvada veebipoes, nii et süsteem oleks võimalikult automatiseeritud ja ei peaks neid lisama käsitsi.

Eesmärk on luua rakendus, mis võimaldab hoida kokku laopinda ja pakkuda klientidele laia sortimenti. Tegemist on ärimudeliga, mille abil saab pakkuda klientidele uusi tootegruppe ilma suurte kulutusteta. See võimaldab ka testida uute toodete või/ja kaubagruppide müügipotentsiaali enne kui need lisada enda lao sortimenti.

3 Probleemi analüüs

Enne arendustöid on oluline teha analüüs, et mõista projekti nõudeid, riske, eesmärke ja ressursse. Analüüs aitab kindlustada, et projekt on hästi planeeritud ja läbimõeldud, mis omakorda aitab vähendada ebaõnnestumise riski ning tagab projekti efektiivsuse ja õigeaegse valmimise.

3.1 Nõuete kogumise tehnika

Ettevõttel ei ole täpselt teada, milline peaks olema soovitud lõpptulemus, kuna rakendus kasvab ajas koos uute tarnijate ja ärinõuetega. Seetõttu otsustati läheneda agiilse metoodikaga.

Rakenduse nõuete väljaselgitamiseks on vaja nõuded kaardistada ja need järjestada tähtsuse järgi. Kuna tegemist on ülesandega, kus meeskonnal varasem tugev kogemus puudub siis, otsustas autor nõuete kogumise tehnikana kasutada ajurünnakut. Ajurünnak on üks levinumatest nõuete kogumise tehnikatest [3]. Antud tehnika õige rakendamine meeskonnas on kõige suurema potentsiaaliga vajalike nõuete leidmiseks ja võimalike väljakutsete avastamiseks [3]. Ülesande planeerimisse kaasatakse nii ettevõtte ostujuhid, kelle üheks ülesandeks on tootesortimendi valik, kui ka e-teenuste arendusjuht, kelle vastutusalasse kuulub tooteinfo haldus.

3.2 Funktsionaalsed nõuded rakendusele

Funktsionaalsed nõuded kirjeldavad tarkvara konkreetseid funktsioone ja omadusi, mis peavad olema realiseeritud, et süsteem saaks täita oma eesmärki. Lihtsamalt öeldes, funktsionaalsed nõuded kirjeldavad mida süsteem peab tegema. [4]

Loodavale rakendusele esitatavad funktsionaalsed nõuded on:

- Rakendus peab võimaldama lisada mitmeid tarnijaid.
- Rakendus peab võimaldama erinevates vormingutes sisendeid (json, xml, csv).
- Rakendus peab võimaldama sisendeid omavahel siduda, ehk mitmest erinevast sisendist ühtset tooteinfot luua.
- Rakendus peab võimaldama erinevaid ühendustüüpe (http, ftp).

- Rakendus peab võimaldama tooteinfo korrektsust kontrollida.
- Tooteinfot peab olema võimalik vastavalt ärinõuetele filtreerida.
- Tooteinfot peab olema võimalik vastavalt ärinõuetele täiendada.
- Rakenduse väljund peab olema standardiseeritud kujul.
- Rakendusele peab võimaldama erinevate vormingutega väljundit (xml, xslx).

3.3 Mittefunktsionaalsed nõuded rakendusele

Mittefunktsionaalsed nõuded kirjeldavad süsteemi kvaliteediomadusi, nagu jõudlus, turvalisus, kasutatavus ja usaldusväärsus. Need nõuded keskenduvad tarkvara üldisele toimimisele ja käitumisele ning sellele, kuidas see vastab teatud standarditele. Lihtsamalt öeldes, need on nõuded mis kirjeldavad kuidas süsteem peaks töötama. [4]

Loodavale rakendusele esitatavad mittefunktsionaalsed nõuded on:

- Süsteem peab olema töökindel.
- Süsteem peab olema kasutajale lihtsasti arusaadav.
- Süsteem peab võimaldama hilisemat laiendust.
- Süsteem peab võimaldama hilisemat uuendust.
- Süsteemi lähtekood peab olema ettevõtte repositooriumis.
- Süsteem peab võimaldama kasutada erinevaid andmebaasimootoreid (MSSQL, MySQL).
- Süsteem peab olema arendatud rakendades kaasaegseid tehnoloogiaid.
- Süsteem peab olema arendatud, nii et kihid oleksid üksteisest eraldatud.
- Süsteem peab võimaldama erinevaid platvorme (Windows, Linux, Docker).
- Süsteem peab töötlema andmeid võimalikult kiiresti.

3.4 Olemasolevate lahenduste võrdlus

Alati ei ole mõistlik hakata rakendust ise ehitama, vaid tasub vaadata, kas on juba nõuetele vastavaid lahendusi loodud, mida saaks kasutusele võtta. Tabelis 1 võrreldakse valmislahenduste ja rätseplahendusi eeliseid ja puuduseid.

Tabel 1. Rakenduse ehitamise ja valmislahenduste võrdlus [5].

	Valmislahendus	**Rätseplahendus**
Litsents ja kontroll	Enamasti antakse kasutuslitsents, tarkvara omandiõigus jääb pakkujale.	Täielik kontroll andmete üle, tarkvara on kasutaja omand.
Funktsionaalsus	Lähtutakse keskmisest ettevõttest, lahendus võib jätta mõned vajalikud funktsionaalsused katmata ja lisaks võib see sisaldada palju lisafunktsionaalsusi, mida tegelikult vaja ei ole ning hakata seetõttu piirama kasutusmugavust.	Luuakse vaid funktsionaalsus, mis on ettevõttele vajalik ja puudub.
Mugavus	Funktsionaalsus on arendatud põhimõttel, et sobiks kõigile. Seetõttu võivad protsessid olla keerukad ja mittearvestada tegelike vajadustega.	Tarkvara luuakse võttes aluseks kasutajate soove ja ettevõtte protsesse.
Kulud	Esialgne investeering on väike, kuid võivad kaasneda pikaajalised kulud (uuendamine, tugi, lisafunktsionaalsuste loomine).	Esialgne investeering on suurem.
Hooldus	Sõltub tootjapoolsest toest, tihti võivad vajalikud muudatused võtta väga kaua aega.	Väga paindlik, võimaldab kiirelt ja paindlikult uuendada, parandada või lisada funktsionaalsust.

Planeeritud rakenduse nõuetele vastavat rakendust autor ei leidnud. Seetõttu vaadeldakse järgnevalt sarnaseid lahendusi, mille kaudu oleks võimalik vajalik funktsionaalsus saavutada.

Veebipood, kuhu antud tooteinfo peab lõpuks jõudma kasutab Magento 2 (nüüdse nimega Adobe Commerce) tarkvara. Magento tarkvarale on loodud mitmeid pistikmooduleid kolmandatelt osapooltelt, mis võimaldavad tooteinfo importimist.

Üheks võimaluseks on Magento moodul Mass Product Import & Update [6], mis võimaldab tooteinfo importi. Moodulis väga hästi lahendatud tooteinfo impordi funktsionaalsus. Puudulik on võimalus enne importi paindlikult tooteid filtreerida. Mooduli kasutamise eelduseks on, et sisendandmed on korrastatud.

Teine võimalus on Magento 2 pistikprogramm Import Products [7], mis võimaldab sarnaselt esimesele tooteinfo importi. Siin tekib täpselt sama probleem - tooteid ei ole

võimalik vastavalt ärinõuetele eelnevalt filtreerida ja mooduli kasutamiseks peab sisendfailide struktuur olema korrektne ja ühtlustatud.

Kolmandana võib välja tuua Pimcore [8] lahenduse, mis on mõeldud tooteinfo halduseks. Tegemist on väga võimeka lahendusega, kus on võimalik enne väljundisse saatmist tooteinfot hallata ja filtreerida. Antud lahendus vajab eraldi arendustöid, et liidestada tarnijate tooteinfoga ja luua sobiv väljund veebipoodi. Selline lahendus sobib pigem loodavale lahendusele lisaks, pakkudes kasutajatele võimalust automaatikaga loodud tooteid täiendavalt tooteinfoga rikastada enne veebipoodi edastatamist.

Kokkuvõtvalt on võrreldavate lahenduste ja rakenduse nõuete vastavus esitatud tabelis 2.

Tabel 2. Olemasolevate lahenduste võrdlus [6]–[8].

Nõue	**Mass Product Import & Update**	**Import Products**	**Pimcore**
Rakendus peab võimaldama lisada mitmeid tarnijaid	+	-	-
Rakendus peab võimaldama erinevates vormingutes sisendeid (json, xml, csv)	+	-	-
Rakendus peab võimaldama sisendeid omavahel siduda	-	-	-
Rakendus peab võimaldama erinevaid ühendustüüpe (http, ftp)	+/-	+/-	+
Rakendus peab võimaldama tooteinfo korrektsust kontrollida	-	-	+
Tooteinfot peab olema võimalik vastavalt ärinõuetele filtreerida	-	-	-
Tooteinfot peab olema võimalik vastavalt ärinõuetele täiendada	-	-	+
Rakenduse väljund peab olema standardiseeritud kujul	+	+	-
Rakendusele peab võimaldama erinevaid väljundeid (xml, xslx)	-	-	-

3.5 Tarkvara arendusmudeli valik

Tarkvara arendusmudeli valik sõltub mitmetest teguritest, näiteks projekti suurus, keerukus ja projekti eesmärgid. Tarkvara arendusmudeleid on võrreldud tabelis 3.

Tabel 3. Koskmudeli ja agiilse mudeli võrdlus [9].

	Koskmudel	**Agiilne mudel**
Projekti ajagraafik	Fikseeritud.	Paindlik ja muutub projekti käigus.
Kliendi kaasamine	Nõuded pannakse paika alguses ja projekti käigus on kliendi kaasamine minimaalne.	Klient on projekti tihedalt kaasatud ja annab projekti käigus tagasisidet
Paindlikkus	Nõuded on eelnevalt kinnitatud ja nende muutmine projekti käigus on keeruline.	Nõuded on paindlikult, projekti käigus on võimalik nõudeid muuta või lisada vastavalt tagasisidele.
Eelarve	Fikseeritud.	Eelarve on paindlik ja keeruline on täpselt hinnata terve projekti maksumust.

Arvestades, et antud projekti nõuded töö jooksul muutuvad või täienevad, pidas autor otstarbekaks kasutada agiilset mudelit.

3.6 Tehnoloogiate valik

Rakenduse loomiseks on olemas mitmeid võimalikke programmeerimiskeeli, raamistikke ning mooduleid. Võttes arvesse püstitatud eesmärki, sellele esitatavaid nõudeid ja lähtetingimusi, valitakse käesolevas peatükis neist kõige sobivaimad rakenduse loomiseks.

3.6.1 Rakenduse tehnoloogia valik

Õige tehnoloogia valimine rakenduse arendamiseks sõltub mitmest tegurist, näiteks projekti eesmärgid, eelarve, ajakava, meeskonna oskused ja kogemused ning rakenduse tulevane laiendatavus [10].

Peamised tegurid õige tehnoloogia valimisel on [11]:

- Kogemus – kasutaja enda piiratud tehnoloogiakogemus on sageli kitsendavam kui kõik muud aspektid. Tehnoloogia valimisel tuleb otsustada, kas eelistada tehnoloogiat, milles on rohkem oskusi, kuid mis ei pruugi olla parim valik, või valida tundmatu tehnoloogia, mis sobib probleemi lahendamiseks paremini. Tundmatu tehnoloogia omandamine võib omakorda pikendada arendusprotsessi.
- Populaarsus ja kogukonna suurus – mida laiem on populaarsus ja kogukond, seda suurem on tõenäosus, et konkreetne tehnoloogia areneb pidevalt ja lahenduste leidmine probleemidele on lihtsam.
- Jätkusuutlikus – tuleb silmas pidada, et suudetakse kaasas käia tehnoloogia pideva arenguga, mis muudab rakenduse haldamise ja uuendamise lihtsamaks.
- Dokumentatsioon – hästi dokumenteeritud tehnoloogiat on lihtsam kasutada ja see annab eeltöö tegemisel selgema arusaama, mida tehnoloogia võimaldab ja mida mitte.
- Turvalisus – ükski tehnoloogia pole täielikult turvaline ja seetõttu tuleb valida tehnoloogiad, mida täiustatakse pidevalt ja eelistada raamistikke, mis pakuvad sisseehitatud lahendusi. Ise loodud lahendused ei ole tõenäoliselt nii turvalised kui raamistikke poolt pakutud.
- Litsents – veebirakenduse kasutamine võib olla piiratud, kui kasutatud tehnoloogiatel on piiravad litsentsid. Probleemide ennetamiseks tuleb veenduda, et antud tehnoloogia litsents lubab veebirakendust kasutada soovitud viisil.

3.6.2 Koodivaramu valik

Koodivaramu kolm kõige populaarsemat lahendust on GitHub, GitLab ja Bitbucket [12].

GitHub on koodivaramu keskkond, milles on üle 330 miljoni repositooriumi ja üle 100 miljoni kasutaja seisuga aprill 2023 [13]. Koodivaramu keskkond on tasuta tingimusel, et andmete maht ei ületa 500 MB [14].

Bitbucket on koodivaramu keskkond, mida kasutab üle 15 miljoni arendaja [15]. Selle tasuta versioon lubab luua piiramatu arvu privaatseid repositooriume ning jagada tööd kuni viie liikme vahel [16]. Bitbucketi loojaks on Atlassian, mille tootevalikus on ka üks

populaarsematest agiilse tarkvara arenduse tööriistadest Jira [17], mille integratsioon Bitbucketiga on suurima funktsionaalsusega võrreldes teiste keskkondadega.

GitLab on koodivaramu keskkond, mis pakub tasuta versioonis 5GB andmemahtu ja kuni viis kasutajat [18]. GitLab keskkond põhineb DevOps arenduskultuuril ning hõlmab ühiskasutatavaid etappe: planeerimine, arendamine, turvamine ja juurutamine [19].

Autori eesmärk koodivaramu kasutuse osas on põhiliselt koodi versioneerimine. Kuna autor töötab rakendusega üksi, siis ei ole vajalikud ühiskasutuse funktsioonid. Kõik vaadeldud koodivaramu keskkonnad võimaldavad vajaminevat funktsionaalsust. Kuna ettevõttes on juba kasutusel Bitbucket ja autoril on seetõttu sellega kõige suurem kogemus, siis võtab autor kasutusele Bitbucketi.

3.6.3 Andmebaasi valik

Andmebaasid on põhimõtteliselt andmete või dokumentide kogumikud, mis kasutavad standardseid meetodeid andmete lisamise, muutmise ja kustutamise jaoks. Peale andmete hoidmise, saab neid algoritmide ja päringute abil otsida ja analüüsida. On olemas mitmesuguseid andmebaasi haldussüsteeme, millest tuntuimad on hierarhilised, relatsioonilised, objektorienteeritud, graafilised, võrgu- ja dokumendipõhised andmebaasid [20]. Üldiselt võib eristada kahte peamist andmebaasi tüüpi: relatsioonilisi ja mitte-relatsioonilisi, mida on võrreldud tabelis 4.

Tabel 4. Relatsioonilise ja mitte-relatsioonilise andmebaasi võrdlus [21]–[23].

	SQL	**NoSQL**
Andmebaasi struktuur	SQL-andmebaasid kasutavad relatsioonilist mudelit, kus andmed salvestatakse tabelitesse. Tabelid on omavahel seotud võtmete abil, mis võimaldab päringute tegemist üle mitme tabeli.	NoSQL-andmebaasid kasutavad erinevaid andmemudeleid, nagu dokumendi-, graafi-, võti-väärtuse- ja lahtiselt veergude põhised mudelid. Need mudelid võimaldavad mitmesuguseid andmestruktuure, nagu JSON-dokumendid, graafid ja tüüpilised võti-väärtuse paarid.

	SQL	**NoSQL**
Päringukeel	SQL-andmebaasid kasutavad standardset SQL-i, mis on hästi dokumenteeritud ja laialdaselt õpetatud.	NoSQL-andmebaasidel puudub ühtne päringukeel, mistõttu erinevad päringud ja süntaksid erinevate andmebaasimootorite vahel.
Skaleeritavus	SQL-andmebaase on raskem horisontaalselt skaleerida, kuna neid tuleb jagada alamandmebaasideks (sharding), mis võib nõuda keerukaid juhtimisprotsesse.	NoSQL-andmebaase on lihtsam horisontaalselt skaleerida, kuna need on loodud algusest peale suurte andmemahtude ja kõrge külastatavusega süsteemide jaoks.
Andmete terviklikkus ja konsistentsus	SQL-andmebaasid kasutavad ACID-omadusi (Atomicity, Consistency, Isolation, Durability), mis tagavad andmete terviklikkuse ja konsistentsuse.	NoSQL-andmebaasid kasutavad sageli BASE-omadusi (Basically Available, Soft state, Eventually consistent), mis pakuvad madalamat andmete konsistentsust, kuid suuremat paindlikkust ja kiirust.

Relatsioonilised andmebaasid on sobivamad, kui olulised on andmete struktureerimine, terviklikkus, ACID omadused ja turvalisus. Mitte-relatsiooniliste andmebaaside eelisteks on horisontaalne skaleeritavus, paindlikus ja kõrge jõudlus. [24]

Kavandatava rakenduse puhul on andmed struktureeritud ja on tähtis andmete terviklikkus, seetõttu on sobivamaks valikuks relatsiooniline andmebaas. Lisaks on relatsioonilises andmebaasis lihtsam teha päringud kasutades standardset SQL päringukeelt. Autoril puudub mitterelatsiooniliste andmebaasidega kogemus, seetõttu oleks selle kasutusele võtmine keerukam ja tuleks arvestada pikema õppimiskõveraga.

Tabelis 5 on võrreldud kõige populaarsemaid relatsioonilisi andmebaase [25].

Tabel 5. Andmebaaside võrdlus [26]–[31].

	Toetatud platvormid	**Litsents**	**Tugevused**	**Nõrkused**	**Kogemus**
Oracle	Windows, Linux, Unix, Solaris	Tasuline (kaubandu	Suure jõudlusega, väga turvaline, suurepärane	Kallis litsentsimine,	Puudub

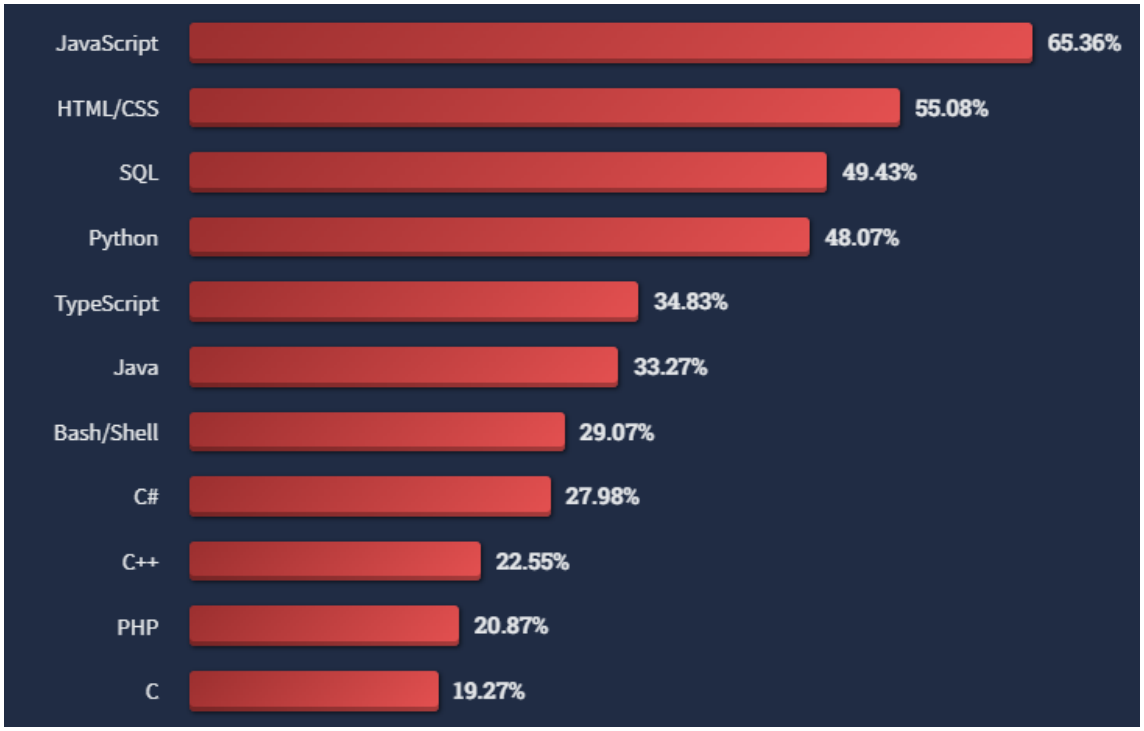
	Toetatud platvormid	**Litsents**	**Tugevused**	**Nõrkused**	**Kogemus**
		slik litsents)	tehniline tugi, ulatuslik funktsionaalsus	keerulisem seadistus	
MSSQL	Windows, Linux	Tasuline (kaubandu slik litsents), Express versioon tasuta (piirangut ega)	Laialdane tugi Microsofti tehnoloogiatele, hea jõudlus ja turvalisus, integreeritud arendusvahendid	Kallis litsentsimine, suurem ressursitarbimine	Väga hea
MySQL	Windows, Linux, Unix, MacOS	Tasuta (avatud lähtekoodi ga; GPL), kaubandu slik litsents (väiksema tele ettevõtetel e)	Laialt kasutatav, hea jõudlus, kergesti õpitav ja seadistatav, palju tööriistu ja ressursse	Suurtes andmebaasides on jõudlus madalam, vähem täiustatud funktsioone	Väga hea
PostgreSQ L	Windows, Linux, Unix, MacOS	Tasuta (avatud lähtekoodi ga; PostgreS QL litsents)	Täielikult ACID-toega, ulatuslik funktsionaalsus, suurepärane jõudlus, hea skaleeruvus	Vähem intuitiivne ja lihtne kui MySQL, nõuab rohkem teadmisi seadistamiseks	Puudub
SQLite	Windows, Linux, Unix, MacOS, Android, iOS	Tasuta (avatud lähtekoodi ga; Public Domain)	Väga kerge, ideaalne väikestele projektidele ja mobiilirakenduste le, lihtne õppida ja kasutada	Ei sobi suuremahulistele projektidele, piiratud konkurentsivõime ja skaleeruvus, vähem turvalisusfunktsio one	Väike

Kõik võrreldavad andmebaasid on sobivad rakenduses kasutamiseks. Oracle kasutamine oleks kallis ja autoril puudub ka sellega kogemus. MSSQL puhul on autoril kogemus

suur, kuid täisversioon on kallis ja Express versiooniga võib tulevikus probleemiks osutuda andmebaasi suuruse limiit 10GB [32]. SQLite kasutamisel võib osutuda probleemseks jõudlus, sest rakenduse andmemaht aja jooksul järjest kasvab, lisaks ei ole SQLitega autoril kogemust. Autor otsustas valida andmebaasiks MySQLi, kuna PostgreSQL kasutusega ei ole autor kokku puutunud.

3.6.4 Programmeerimiskeele valik

Kõige rohkem kasutatavad programmeerimiskeeled Stackoverflow 2022 küsitluse põhjal on JavaScript, Python, Java, C#, C++ ja PHP [33], mida on näidatud ka joonisel 1. Programmeerimiskeele valikust sõltub ka raamistike valik ehk enne keele valimist tuleb kindlasti tutvuda ka potentsiaalsete raamistikega.



Joonis 1. Programeerimiskeelte populaarsus 2022 [33].

PHP – üldkasutatav skriptimiskeel, mida kasutatakse serveripoolsetes lahendustes dünaamiliste veebilehtede ja veebirakenduste loomiseks. Keelt peetakse algajatele sobivaks ning tihti on see ka algajatel esimeseks keeleks, mida õpitakse. [34]

Java – üldkasutatav objektorienteeritud programmeerimiskeel, arendatud Sun Microsystems poolt. Java on avatud lähtekoodiga keel, mis on platvormist sõltumatu.

Java on sobiv valik keeruliste projektide tegemiseks. Keelt peetakse keeruliseks õppida. [34]

JavaScript – objektorienteeritud skriptimiskeel, mis võimaldab objektorienteeritud, protseduurilist või funktsionaalset programmeerimist. JavaScript on populaarseim programmeerimiskeel aastal 2022 ning valdav enamik populaarsemaid veebilehekülgi kasutavad keelt kliendipoolselt. [34]

C# – üldkasutatav objektorienteeritud programmeerimiskeel, arendatud spetsiaalselt Microsofti poolt .NET raamistiku jaoks. C# sobib nii Windows rakenduste, mängude, kui ka veebirakenduste arenduseks. C# on sobiv keerukamate rakenduste jaoks. [34]

Python – üldkasutatav programmeerimiskeel, mis võimaldab aspektipõhist, objektorienteeritud või funktsionaalset programmeerimist. Keelt on kritiseeritud suhteliselt madala täitmiskiiruse ja selle meetodite pikkade definitsioonide tõttu. [34]

C++ – üldkasutatav masinorienteeritud programmeerimiskeel, mis põhineb C-l, mis on üks vanimaid programmeerimiskeeli. Keele suurimad tugevused on selle erinevad kombinatsioonid ja efektiivne masinale orienteeritud programmeerimine. Keel on tänapäeval üks populaarsemaid programmeerimiskeeli süsteemi- ja rakendusprogrammeerimises. [34]

Tabelis 6 on võrreldud ülalnimetatud programmeerimiskeeli, võrdlemisel on aluseks võetud autori kogemus ja õppimiskeerukus.

Tabel 6. Programmeerimiskeelte võrdlus.

Keel	**Kogemus**	**Õppimiskeerukus**
PHP	Hea	Madal [35]
Java	Halb	Keskmine [35]
Javascript	Keskmine	Madal [35]
C#	Väga hea	Keskmine [35]
Python	Keskmine	Madal [35]

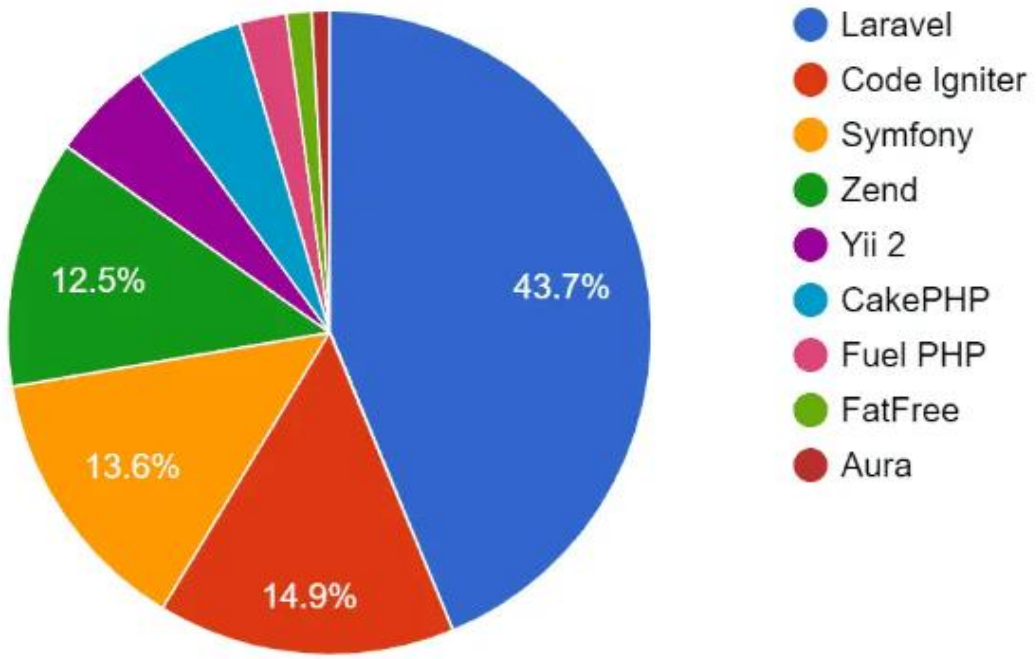
Keel	Kogemus	Õppimiskeerukus
C++	Puudub	Keskmine [35]

Arvestades, et iga uue keele õppimine võtab aega, oleks mõistlik valida selline keel, kus autoril on rohkem kogemust ja rakenduse arendusaeg seetõttu ka lühem. Nendeks keelteks oleksid C# ja PHP, mida autor kasutab ka oma igapäevases töös. C# keelega on autoril suurem kogemus ja seetõttu ei ole oluline ka keele suurem õppimiskeerukus. PHP samas on madala õppimiskeerukusega, nii et puuduvad teadmised on kiirelt omandatavad.

3.6.5 Raamistiku valik

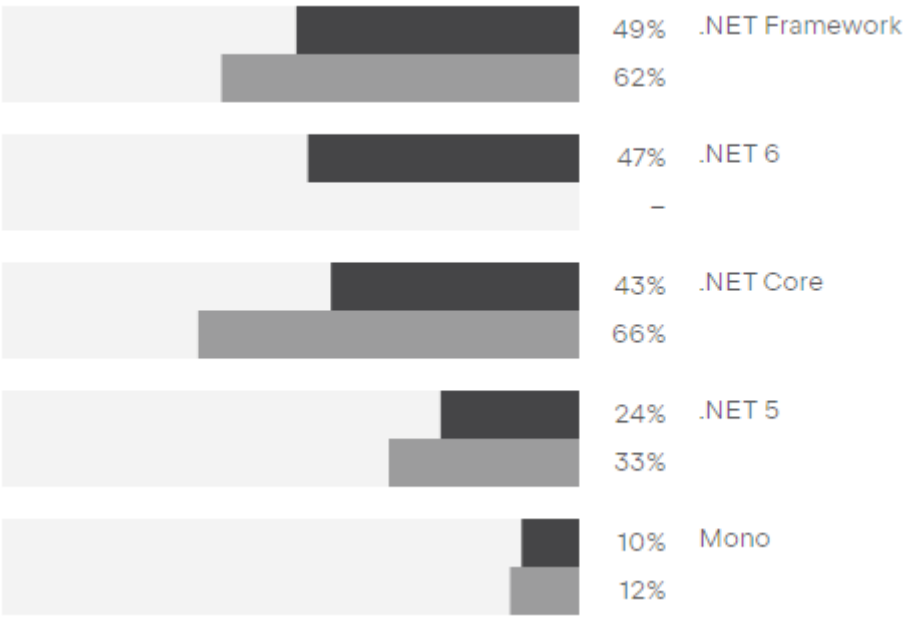
Raamistik on teekide ja tööriistade kogum, mis aitab arendajatel luua konkreetsele probleemile lahendusi. Raamistikud võimaldavad arendajatel keskenduda probleemi põhiaspektidele, pakkudes struktuuri ja ühiseid komponente, mis hõlbustavad rakenduse loomist ja hooldamist.

Jooniselt 2 selgub, et programmeerimiskeele PHP kõige populaarsem raamistik on Laravel .



Joonis 2. PHP raamistike võrdlus [36].

C# programmeerimiskeeles on kõige populaarsem raamistik on .NET, endise nimega .NET Core [37] ja selle erinevad versioonid (vt joonis 3).



Joonis 3. C# raamistike võrdlus [37].

Laraveli raamistik on mõeldud eelkõige veebirakenduste arenduseks [38]. Kuna käesolev rakendus on mõeldud töötama ilma eesrakenduseta, siis pole Laravel raamistik kõige sobivam. Küll aga on loodud Laraveli komponentide peale mikroraamistik Laravel Zero [39], mis on mõeldud just konsoolirakenduste arenduseks.

.NET raamistik ei sisalda eesrakenduse arenduskomponente [40], seetõttu oleks selle kasutuselevõtt arvestades projekti nõudeid sobiv.

Raamistik tuleb valida ka selle järgi, et see võimaldab lahendada etteantud probleemi, kuid sisaldab võimalikult vähe projektis mittevajaminevaid mooduleid ja funktsioone, mis võivad tekitada jõudlus- ja turvaprobleeme. Kuna väga suurt osa Laraveli raamistiku funktsionaalsusest ei võeta käesolevas rakenduses kasutusele, pole Laravel raamistiku kasutamine autori hinnangul otstarbekas. Lisaks autoril puudub selle raamistikuga kogemus ja seetõttu otsustab autor kasutada .NET raamistikku.

3.6.1 Tööriistade valik

.NET rakenduste arendamiseks on kõige rohkem levinud järgmise kolme integreeritud arenduskeskkonna kasutamine: JetBrains Rider, Visual Studio ja Visual Studio Code [41].

Rider loodi JetBrains-i poolt aastal 2016 ning seda on siiani arendatud. Rider toetab Linux, Windows ja MacOS operatsioonsüsteeme. Rider arenduskeskkonna kasutamine on tasuline, kuid tudengile on arenduskeskkonna kasutamine tasuta. Rideril on sisseehitatud kompilaator ReSharper, mis võimaldab vigu varakult tuvastada ning pakub võimalusi neid kiiresti parandada. [42]

Visual Studio on Microsofti poolt loodud integreeritud arenduskeskkond, mis on mõeldud peamiselt Microsofti tehnoloogiate jaoks, kuid toetab ka paljusid teisi programmeerimiskeeli. Visual Studio on saadaval Windowsi ja MacOSi operatsioonsüsteemidele. Visual Studio on kommertstarkvara, kuigi sellel on ka tasuta versioon Visual Studio Community Edition. [43]

Visual Studio Code (tuntud ka kui VSCode) on tasuta, avatud lähtekoodiga koodiredaktor, mille on loonud Microsoft. See on saadaval Windowsi, MacOSi ja Linuxi platvormidele. Visual Studio Code on mõeldud peamiselt veebiarenduse jaoks, pakkudes tuge erinevatele programmeerimis- ja skriptimiskeeltele nagu JavaScript, TypeScript, Python, C#, PHP, Go ja paljud teised. Visual Studio Code on tuntud oma laiendatavuse ja kohandatavuse poolest. Kasutajad saavad lisada laiendusi, mis pakuvad täiendavaid funktsioone, nagu keelespetsiifiline süntaksi esiletõstmine, koodi täiendamine (IntelliSense), veaparandus , versioonikontrolli integratsioon, koodi ümberkorraldamine, testimisvahendid ja palju muud. [44]

Kõik kirjeldatud integreeritud arenduskeskkonnad on sobivad rakenduse loomiseks. Küll aga pakub Visual Studio Code vähem sisseehitatud funktsionaalsust ja arenduseks on vaja hakata sobivad laiendusi otsima. Kuna laienduste valik on suur, võib kuluda palju aega sobivate laienduste leidmiseks. Valikusse jäävad Visual Studio ja Rider. Autor otsustas valida Rideri, tuginedes oma suuremale kogemusele selles keskkonnas ning soovile vältida uue keskkonna õppimisega seotud ajakulu.

3.6.2 Eesrakenduse valik

Käesoleva töö raames loodavale rakendusele eesrakendust ei planeerita. Eesrakenduse planeerimine oleks keeruline, kuna algses faasis ei suudetud välja töötada nõudeid, mis probleemi peaks eesrakendus lahendama. Arvestades ka seda, et rakendus arendatakse iteratsioonide kaupa ja on suur tõenäosus, et nõuded lisanduvad või muutuvad, tuleb eesrakenduse juurde tagasi pöörduda siis, kui on võimalik sõnastada konkreetsed nõuded ja vajadused.

3.6.3 Rakenduse haldus

Rakendus on planeeritud töötama sisevõrgus ja ei ole avalikkusele kättesaadav. Seetõttu otsustatakse antud rakendus paigalda sisevõrku, olemasoleva serveri peale. Hetkel töötab kogu ettevõtte serveripark virtualiseeritud serveritel. Kuna ettevõttes on juba kasutusel Windowsi serverid, mis ei vaja täiendavaid kulutusi, osutus valituks virtualiseeritud server, mis kasutab Windowsi operatsioonisüsteemi. Testiti ka Dockeri võimalust, kuid Dockeri lisandumine tähendaks suuremat keerukust serveripargi halduses, sest tegu oleks uue tehnoloogiaga, mis leiaks hetkel kasutust vaid ühe rakenduse jaoks.

3.7 Analüüsi kokkuvõte

Analüüsi käigus anti ülevaade loodava rakenduse funktsionaalsetest ja mittefunktsionaalsetest nõuetest, vaadeldi ka erinevaid tehnilisi võimalusi rakenduse arendamiseks ning nende eeliseid ja puudusi.

Rakenduse andmeid otsustati hoida relatsioonilise mudeliga andmebaasis. Andmebaasisüsteemiks valis autor MySQLi, kuna see on vabavaraline ja sobib kasutamiseks erinevate teenuspoolsete tehnoloogiatega. Lisaks on sellel hea jõudlus ja ta sobib töötamiseks ka suurte andmemahtudega.

Rakenduse keeleks valiti C# koos .NET raamistikuga, kuna sellega on lõputöö autoril hea kogemus ja suur osa ettevõttes olevaid lahendusi on loodud just .NET raamistikku kasutades. Lisaks on keel hästi dokumenteeritud.

Koodivaramu keskkonnaks valiti Bitbucket, mis täidab koodi versioneerimise nõuded ja on juba ka ettevõttes varasemalt kasutusel.

Kuna rakendus on mõeldud kasutamiseks sisevõrgus, siis valiti lokaalne serverilahendus. Serverid võivad olla nii Windowsi või Linux operatsioonsüsteemiga, kuid antud rakenduse jaoks valiti olemasolev Windowsi server. Kood kirjutatakse JetBrains Rider arenduskeskkonda kasutades.

4 Rakenduse loomine

Rakendus valmib etappide kaupa kasutades agiilset metoodikat, alustades sisendandmete ühtlustamisest ja lõpetades toimiva lahendusega. Rakenduse arendus on jaotatud nelja suuremasse peatükki: andmebaasi arendus, rakenduse arendus, rakenduse väljundi arendus ja testimine. Rakenduse loomine jagatakse väiksemateks etappideks ehk iteratsioonideks. Iga iteratsiooniga luuakse lahendusele juurde loogiline osa funktsionaalsusest ja iga etapi lõpus toimub manuaalne testimine.

4.1 Esimene iteratsioon – andmete ühtlustamine

Esimese etapi käigus valitakse välja kolm tarnijat, analüüsitakse tarnijate tooteinfo voog ja selle põhjal koostatakse sobiv andmemudel.

4.1.1 Tarnijate valik

Tarnijate valiku kriteeriumiteks on:

- **Tarnija tootevalik**. Tarnijal peab olema piisavalt lai ja mitmekülgne tootevalik, et liidestuse loomisel veebipoega oleks võimalik mõista selle ärilist kasutegurit. Kui valida tooted, mille vastu kliendil huvi ei ole või tarnija, kellel on väike tootevalik, siis võib tekkida olukord, et kuigi tooted on veebipoes olemas, ei kajastu äriline kasu ettevõtte müüginumbrites.
- **Tarnijaga koostöö**. Tarnijaga peab olema juba toimiv koostöö, et rakenduse kasutamine ei takerduks näiteks ostuhindade kokkuleppimise, kaupade transpordi vm peale.
- **Tarnijate tooteinfo**. Otstarbekas on valida võimalikult erineva struktuuriga tooteinfo, lihtsustamaks arusaamist, milline peab olema loodav lahendus, et oleks võimalik luua andmemudel, mis sobitub erinevate tarnijate tooteinfo andmete struktuuriga. Kui valida väga sarnase kujuga tooteinfo, siis võib tekkida risk, et andmemudel on liiga spetsiifiline ja ei pruugi sobituda järgnevate tarnijate tooteinfoga.

Peale tarnijate väljavalikut kaardistatakse sissetulev tooteinfot ja hakatakse selle põhjal looma rakenduse andmemudelit. Lähtutakse printsiibist, et mudel tuleb koostada vähima tooteinfoga andmefaili järgi. Kui võtta aluseks kõige täiuslikum tooteinfo, siis võib

tekkida probleem, et tulevikus mõnel tarnijal ei ole kõiki kohustuslikke andmeid ja seetõttu tekib andmetes ebakõla.

Algselt kasutatakse andmete pärimiseks Postman rakendust, millega on võimalik pöörduda tarnijate APIde (Application Programming Interfaces) poole ja infot pärida. Postman on tarkvaraarendajatele suunatud rakendus, mis aitab arendada, testida, dokumenteerida ja jälgida API-sid. See on populaarne tööriist, mis lihtsustab API-ga suhtlemist, võimaldades kasutajatel luua ja saata erinevaid HTTP päringuid ja analüüsida saadud vastuseid. [45], [46]

4.1.2 Toote identifikaator

Igal tootel peab olema unikaalne tunnus, mille järgi seda eristada teistest toodetest. Rahvusvaheliselt on kokku lepitud EAN standardis [47]. Paraku selgub, et mitte kõik tarnijad ei pane oma tooteinfoga kaasa EAN koodi, nii et selle kasutamine kahjuks ei ole võimalik. Teise alternatiivina vaadatakse tootjakoodi (tootja poolt loodud unikaalne tunnus). See kood on kõikidel toodetel kaasas ja seetõttu otsustatakse see kasutusele võtta.

4.1.3 Andmete struktuur.

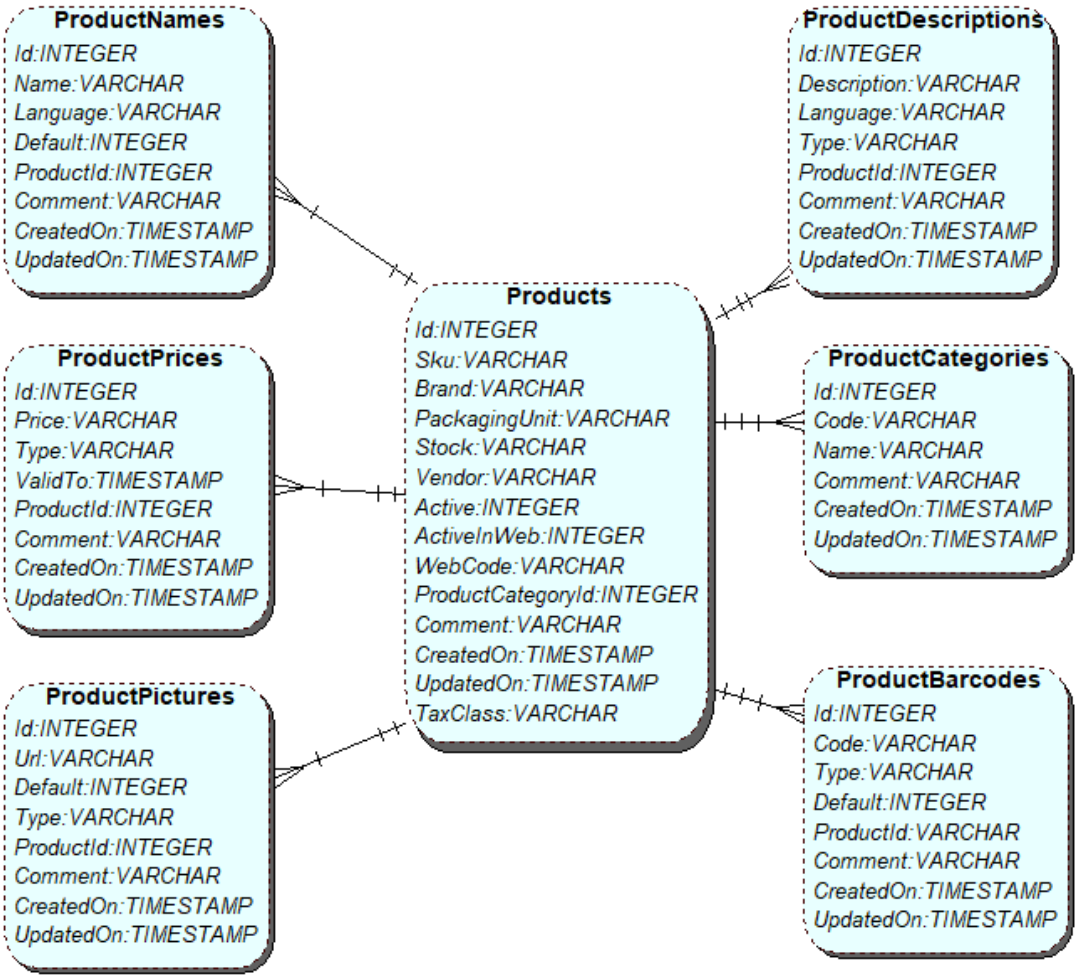
Peale tarnijate tooteinfo failide analüüsi selgub, et andmefailid on väga erinevad ja ühtset loogikat on keeruline luua. Seetõttu võetakse vastu otsus, et tarnijate poolt esitatud infost kaasatakse ainult baasinfo ja toodete lisainfot hakatakse pärima tooterikastamise platvormi Icecat vahendusel. Icecat pakub toodetele pilte, atribuute jm infot ühtlustatud kujul. Icecatil on kaks versiooni, tasuta ja tasuline. Tasuta versioonis on ainult tooteinfo mida on tootjad ise Icecati lisanud, tasulises versioonis on ka ligipääs Icecat enda andmebaasile. [48]

Ettevõte otsustas kasutusele võtta tasulise Icecat versiooni, kuna selle andmebaas katab kõige suuremat hulka tooteid ning seetõttu on võimalik veebipoe toodetele maksimaalselt tooteinfot lisada.

4.1.4 Andmebaasi mudeli loomine

Peale andmete kaardistamist luuakse olemisuhte diagramm. Autor loob olemisuhte diagrammi selliselt, et see oleks võimalikult lihtne, kuid võimaldaks tulevikus ärinõuete

lisandumisel uusi tabeleid või välju lisada. Olemisuhte diagramm luuakse Qsee [49] tarkvaraga ja on esitatud joonisel 4. Olemisuhte diagrammi kesksel kohal on toote baasinfo tabel ja sealsed andmeväljad on sellised, mida alati on ühe toote kohta üks. Olemisuhte diagrammil välja toodud teised tabelid on üks-mitmele seoses toote baasinfo tabeliga, kuna neid andmed võib olla ühe toote kohta mitu.

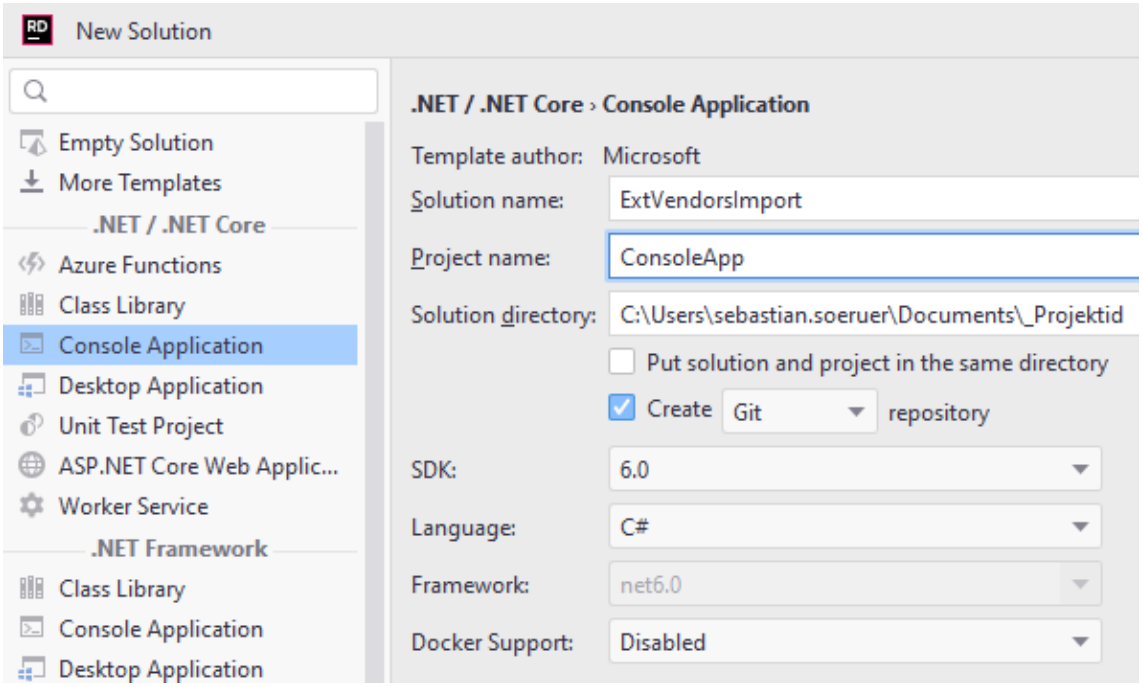


Joonis 4. Olemisuhte diagramm.

4.2 Teine iteratsioon – rakendus

Teises arendusetapis eesmärk on on luua rakenduse esmane versioon, mis suudab andmeid alla laadida, teostada esmase filtreerimise ja salvestada tooteinfo andmebaasi.

Rakendus luuakse kasutades Jetbrains Riderit. Rakendus luuakse konsoolirakendusena, kuna eesrakenduse järgi hetkel vajadust ei ole. Projekt lisatakse automaatselt git versioonihaldusesse. Rakenduse loomise seadistus on välja toodud joonisel 5.



Joonis 5. .NET rakenduse loomine.

Järgmise sammuna valmistatakse ette MySQL andmebaas. Andmebaasi luuakse rakenduse tarbeks kasutaja ja antud info salvestatakse rakenduse konfiguratsioonifaili, ühenduse loomiseks vajalik ühendusstring on toodud välja joonisel 6.

```
„MYSQL“: „server=xxx.xxx.xxx.xxx; port=3306; database=ExtVendorsImportV1;
user=xxxx; password=xxx; Persist Security Info=False; Connect
Timeout=300;CharSet=utf8mb4;“
```

Joonis 6. Andmebaasi ühendamine.

Edasi valmistatakse ette projekti baasstruktuur, kasutades DDD (Domain Driven Design) lähenemist ja lisatakse vajalikud moodulid andmebaasiga suhtluseks, XML failide koostamiseks ja andmete valideerimiseks.

Peamised moodulid, mis projektis kasutusele võetakse on välja toodud tabelis 7.

Tabel 7. Projektis kasutatavad moodulid.

Moodul	**Selgitus**
ClosedXML	Kasutatakse XML failide lugemisel ja kirjutamisel.
Dapper	Kasutatakse MySQL andmebaaside poole pöördumisel.

Moodul	**Selgitus**
Entity Framework	Kasutatakse rakenduse andmebaasi poole pöördumisel.
Magic.NET	Piltide redigeerimismoodul, kasutuses pildifaili korrektsuse valideerimisel.
FluentValidation	Kasutatakse andmete korrektsuse valideerimisel.
xUnit	Ühiktestide loomise moodul.
SSH.NET	Kasutatakse veebipoega ühenduse loomisel ja sinna käskude saatmisel.
CsvHelper	Kasutatakse csv failide lugemisel.
Newtonsoft.Json	Kasutatakse json failide lugemisel ja kirjutamisel.

Järgmisena luuakse tarnijapõhised repositooriumid ja funktsioonid tooteandmete küsimiseks. Joonisel 7 on esitatud tooteinfo päringu funktsioon.

```
public ICollection<WebProduct> GetProducts()
        {
            _httpClient.DefaultRequestHeaders.Authorization =
                new AuthenticationHeaderValue("Bearer",
                    _jwt);
            var response =
_httpClient.GetAsync("Catalog/Products").Result;
            response.EnsureSuccessStatusCode();
            var content = response.Content.ReadAsStringAsync().Result;

            var resp =
JsonConvert.DeserializeObject<List<Item>>(content);

            if (resp == null) return new List<WebProduct>();

            return resp.Select(i => new WebProduct
            {
                Sku = i.Id,
                Mpn = i.ManufacturerCode,
                Ean = i.EanCode,
                Brand = i.VendorName,
                PackagingUnit = null,
                ProductName = i.Name,
                ProductDescription = i.FullDsc,
                ProductPrice = i.Price,
                ProductPicture = i.ImagePath,
                ProductStock = i.Quantity,
                ProductCategory = i.CatalogName
            }).ToList();
        }
```

Joonis 7. Tooteinfo küsimine.

Viimase sammuna luuakse tooteinfo valideerimiskiht, mille ülesandeks on eemaldada puuduva või ebakorrektse baasinfoga tooted, peale mida salvestatakse tooted andmebaasi.

4.3 Kolmas iteratsioon – äriloogika kiht

Kolmandas arendusetapis luuakse äriloogika kiht, mille ülesandeks on filtreerida või muuta tooteinfot vastavalt nõuetele. Etapi lõpuks on olemas toimiv äriloogika kiht, mille alusel saab filtreerida tooted enne nende veebipoodi edastamist.

Esmalt tuleb valida, kas filtreerimine ja muutmine hakkab toimuma toote- või tootekategooria põhiselt. Esimese lähenemise puhul oleks võimalik saavutada täpsem ja detailsem tulemus, näiteks iga toodet on võimalik eraldi kuvada või mitte kuvada, kuid siin tekib vajadus hakata seda protsessi manuaalselt kontrollima. Iga toode tuleb lisada

käsitsi lubatuks. Kui tooteid ei lisanduks, oleks selline lähenemine teostatav, kuid kuna toodete valik on pidevas muutumises, siis oleks halduskoormus sellise lähenemise puhul liiga suur.

Teine lähenemine oleks võtta aluseks tootekategooriapõhine valik, ehk vaadatakse tooteid nende kategooriasse kuuluvusse järgi. Tootekategooriaid on oluliselt vähem ja seetõttu on ka seda protsessi kergem hallata. Lisaks kaob ka uute toodete lisandumisel vajadus need käsitsi lubatuks märkida. Kui tootekategooriasse ilmub uus toode, siis automaatselt on ka antud toode lubatud või keelatud, vastavalt tema kategooriale. Kuna eesmärgiks on automaatne rakendus, kus oleks võimalikult vähe vaja kasutaja sekkumist, siis kasutatakse tootekategooria põhist valikut.

Esimene toodete filtreerimine toimub juba tooteinfo impordil, enne kui neid andmebaasi salvestatakse. Selles etapis vaadatakse üle, et kokkulepitud tooteinfo oleks tootel olemas, kui see on puudu siis antud toodet andmebaasi ei salvestada. Näiteks kui tootel on puudu tootekood, siis sellise info andmebaasi salvestamine rikuks nõuet, et igal tootel peab olema unikaalne identifikaator.

Teisene toodete filtreerimine ja muutmine toimub juba vastavalt kokkulepitud ärinõuetele. Kuna on valitud on tootekategooriapõhine lähenemine, siis selles kategoorias võib olla tooteid, mida ei soovita veebipoes näidata.

Selle tarbeks luuakse BLL.APP kihti uus teenus, *ProductFilterService*. Iga toote olem läbib filtreerimisteenuse. Tabelis 8 on välja toodud kokkulepitud filtrid.

Tabel 8. Tooteinfo filtrid.

Filter	**Kirjeldus**
PRICELESSERFILTER	Kui toode kuulub määratud kategooriasse ja hind on väiksem kui limiit, siis toodet veebipoodi ei edastata.
PRICEGREATERFILTER	Kui toode kuulub määratud kategooriasse ja hind on suurem kui limiit, siis toodet veebipoodi ei edastata.
BRANDFILTERINCLUDE	Filtreeritakse välja tooted, mis kuuluvad määratud kategooriasse ja mille bränd on valimis.

Filter	Kirjeldus
BRANDFILTEREXCLUDE	Filtreeritakse välja tooted, mis kuuluvad määratud kategooriasse ja mille bränd ei ole valimis.
BRANDFILTEREXCLUDEBYVENDOR	Filtreeritakse välja tooted, mis kuuluvad määratud kategooriasse,on kindla tarnija tooted ja mille bränd ei ole valimis.
EXTRACHARGE	Võimaldab tootekategoorias olevatele toodetele määrata kindlaks määratud lisatasu (nt tühja kasseti tasu).
NAMEFILTERINCLUDE	Kui toode kuulub määratud kategooriasse ja nimes esineb määratud sõnaosa, siis toode edastatakse veebipoodi.
NAMEFILTEREXCLUDE	Kui toode kuulub määratud kategooriasse ja nimes esineb määratud sõnaosa, siis toodet ei edastata veebipoodi.
MANUFACTURERCODE	Kui tootjakoodis on määratud mingi kindel sõnaosa, siis toodet ei edastata veebipoodi (näiteks „Damaged“).

Lisaks eelnevalt välja toodud filtreerimissüsteemile on samas teenuses määratud ka võimalused tooteinfo teisendamiseks. Tooteinfo teisendajad ja nende kirjeldused on esitatud tabelis 9.

Tabel 9. Tooteinfo teisendajad.

Teisendaja	Kirjeldus
SetStockZeroWhenNotInMultiImport	Kui toodet mingil põhjusel enam ei saadeta tarnija poolt, siis määratakse antud toote laoseis nulliks.
SetStockZeroWhenStockIsNegative	Kui tootel on negatiivne laoseis, siis määratakse toote laoseisuks null.
RemoveAttributesWhenUpdateIsDisabled	Kui toode on märgitud mitte uuendatavaks veebipoes, siis antud tootel enam tooteinfot peale hinna ja laoseisu ei uuendata.
TranslateCategories	Kui toodet on vaja määrata mõnda teise kategooriasse, kui tarnija selle määranud on.
AddStaticPrice	Olenemata toote tarnija poolsest hinnast on tootel alati määratud hind.

Teisendaja	Kirjeldus
BrandTranslation	Kui on vaja muuta tarnija poolt määratud toote bränd (näiteks HP Inc muudetakse HP).

Tooteinfo filtreid ja teisendajaid hoitakse json failides, kus kirjeldatakse vajalikud parameetrid filtreerimis- ja teisendamisüsteemile, nagu on näitatud joonisel 8. Lisaks on joonisel 9 esitatud näide tooteinfo filtreerimisfunktsioonist.

```
{
    „Name“ : „Elmaks, Jura, Blendtec, Beko, Miele, Apple, Pexman“,
    „Comment“ : „2022-08-15. Beko välistus“,
    „Vendor“ : „Vendor X“,
    „Type“ : „BrandFilterExcludeByVendor“,
    „Category“ : „“,
    „Brands“ : [„Jura“, „Blendtec“, „Beko“, „Miele“, „Apple“,
„Pexman“]
}
```

Joonis 8. Tooteinfo filtreerimise näidis.

```
private FilterServiceDto FilterByBrandExcludeByVendor(FilterServiceDto
productDto, FilterDto filter)
    {
        if (filter.Brands == null || filter.Brands.Count < 1) return
        productDto;
        if (filter.Brands.Contains(productDto.Product.Brand,
        StringComparer.OrdinalIgnoreCase))
        {
            productDto.IsFilteredOut = true;
            productDto.Filters!.Add(
            System.Reflection.MethodBase.GetCurrentMethod()!.Name);
        }
        return productDto;
    }
```

Joonis 9. Tooteinfo filtreerimisfunktsioon.

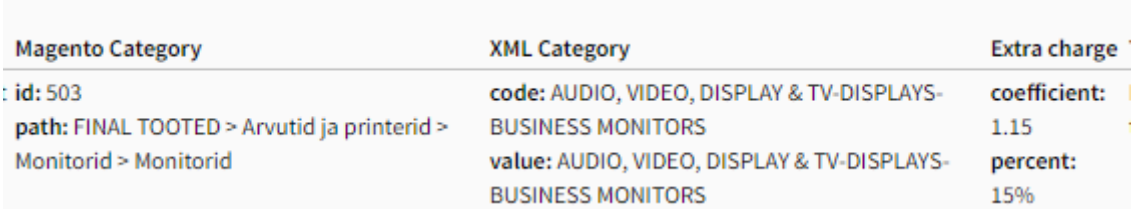
4.4 Neljas iteratsioon – tooteinfo väljund

Neljandas etapis luuakse rakendusele väljundid tooteinfo edastamiseks veebipoodi.

Veebipoel on juba olemas võimekus importida tooteinfot läbi ettevõttes asuva FTP serveri kindla XML struktuuriga failist. See lahendus loodi ettevõtte arenduspartneri

poolt varasemalt võimaldamaks sarnast funktsionaalsust kui käesolev rakendus. Antud lahendusest loobuti, kuna see lahendus ei täitnud kõiki seatud nõudeid ja oli liiga keeruline. Lisaks oleks kõikide nõuete täitmine tekitanud veebipoodi liialt suure keerukuse, mis oleks hakanud avaldama mõju ka veebipoe jõudlusele. Kuna antud lahenduse tooteinfo impordi funktsionaalsus oli sobiv ka loodavas rakenduses kasutamiseks, siis otsustati see osa veebipoes kasutusele võtta. See määras ka nõuded käesoleva rakenduse väljundile, mis peab vastama juba olemasolevale impordilahendusele.

Veebipoe ülesanne on lugeda sisse etteantud XML fail, määrata tooted veebipoes olevatesse kategooriatesse, lisada määratud juurdehindlus ning luua või uuendata tootekaart, mida on näidatud joonisel 10.



Joonis 10. Veebipoe kategooriate ühendamine.

Kuna rakendusele ei looda rakendusliidest, siis üheks lisaväljundiks luuakse Exceli tabel, mis salvestatakse ettevõtte ühiskausta. Sealt on võimalik kasutajatel kontrollida, mis tooteinfo ja mis kujul saadeti veebipoodi. Lisaks saavad antud tabeleid kasutada ka teised töötajad et kontrollida tarnija tootevalikut, kuna veebipoes ei ole kõik tooted nähtaval.

4.5 Testimine

Põhiline rakenduse testimine toimub arenduse jooksul manuaalselt, kuna tarnijate tooteinfo andmetes on keeruline andmete korrektsust testida. Enne veebipoodi jõudmist läbivad andmed mitmeid kihte ja kõiki neid etappe on keeruline automaattestidega katta.

Testijateks valitakse ettevõtte töötajad, kuhu kuuluvad inimesed IT osakonnast, kellel on rohkem tehnilisi teadmisi ja ostujuhid, kelle igapäevaseks tööks on toodete sortimendi haldus.

Lisaks manuaalsele testimisele kasutatakse äriloogika kihi testimiseks üksusteste. .NET rakenduste testimiseks on Microsofti dokumentatsiooni järgi kolm põhilist tööriista NUnit, MSTest ja xUnit [50]. Nendest kõige rohkem soovitatakse kasutada tööriista xUnitit, kuna see on teistest rohkem laiendatav, vähemate atribuutidega, mistõttu kood on puhtam ja kergem hooldada [51]. Kuna lisaks on ka autoril pikaajaline kogemus xUniti kasutusel, siis kirjutatakse testid tööriista xUnitit kasutades.

Testide eesmärk on kontrollida toote filtreerimislahendust, ehk kas tooteinfo muutub vastavalt seadistatud filtritele. Testi näide on välja toodud joonisel 11.

```
[Fact]
public void TestStaticPriceFilter()
{
    var filterService = new ProductFilterService(new FilterRepositoryTest(),
new WebItemRepository());
    var newPrice = filterService.FilterProducts(prodRepo.GetProducts(),
"acc")
        .First(p => p.Product.Sku.Equals("T0019"))
        .Product.ProductPrices.First().Price;

    Assert.Equal(34.35m, newPrice);
}
```

Joonis 11. Filtreerimisfunktsiooni test.

Manuaalne testimine toimub regulaarselt rakenduse loomise käigus, kus võrreldakse tarnijate veebilehelt saadud infot ettevõtte veebipoes olevaga. See võimaldab saada aru, kas tooteinfo on liikunud korrektselt läbi kõikide kihtide. Automaattestid käivitatakse iga uue funktsionaalsuse lisandumisel, et veenduda toodete filtreerimislahenduse töö korrektsuses. Kuna tegemist on keeruka süsteemiga, siis manuaalne testimine oleks liiga aeganõudev ja suureneb risk, et testija ei näe kõiki võimalikke vigu rakenduses.

5 Valmis rakenduse kirjeldus

Käesolevas peatükis kirjeldatakse valminud rakenduse arhitektuuri, struktuuri ja suurimaid väljakutseid, mis esines tarnijate tooteinfo liidestamisel ning võimalikke edasiarendusi.

5.1 Rakenduse arhitektuur

Selle töö rakenduse arhitektuur põhineb SOLID-põhimõtetel [52], mille kohaselt peab tarkvaraarendus järgima järgmisi viit põhimõtet:

S - Üksikvastutuse põhimõte (Single Responsibility principle) tähendab, et klassil või moodulil peaks olema ainult üks põhjus muutumiseks. See tähendab, et klasside funktsioonid peaksid olema selgelt piiritletud, nii et ühe muudatuse tegemine ei häiriks mitut erinevat äriloogilist protsessi.

O - Avatud-suletud põhimõte (Open-Closed principle) näeb ette, et tarkvaraüksust peaks saama vabalt laiendada, kuid mitte muuta, kuna see võib häirida süsteemi teiste osade toimimist.

L - Liskovi asenduspõhimõte (Liskov Substitution principle) tähendab, et programmi objektid peaksid olema asendatavad nende alamtüüpidega.

I - Liideste eraldamise põhimõte (Interface Segregation principle) eelistab mitut spetsiifilist liidest ühele üldkasutatavale liidesele.

D - Pöördsõltuvuse põhimõte (Dependency Inversion principle) nõuab, et tuginetakse abstraktsioonidele ning kõrgema taseme moodulid ei tohiks sõltuda madalama taseme moodulitest.

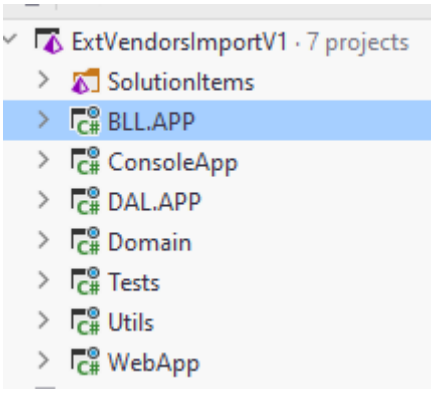
Lähtudes neist põhimõtetest koosneb arendatava rakenduse arhitektuur järgmistest kihtidest:

- Domeen: andmemudeli kiht, mis koondab lihtsaid andmeklasse ilma äriloogikata.
- DAL (Data Access Layer): andmevahetuse kiht, mis vahendab suhtlust äriloogika ja andmebaasi vahel.

- BLL (Business Logic Layer): rakenduse äriloogika kiht, kus teenuste abil koondatakse veebirakenduste kontrollerite jaoks vajalikud andmed.
- ConsoleApp: rakenduse kiht.

5.2 Valmis rakenduse struktuur

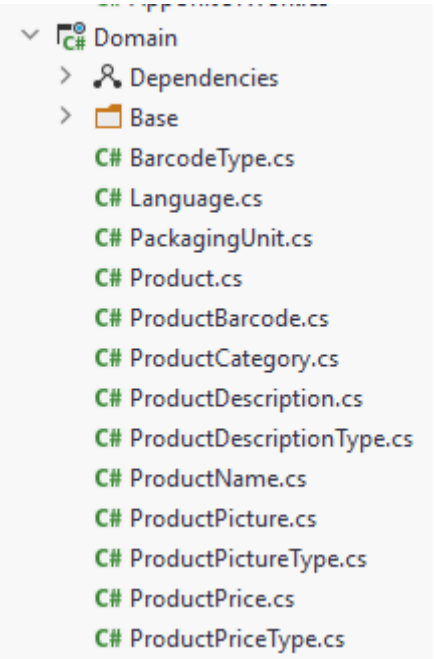
Rakendus luuakse kasutades kihilist struktuuri ja liideseid, mis võimaldab tulevikus teha muudatusi ühe taseme piires nii, et ei oleks vajadust teisi ümber kirjutada. Lisaks lihtsustab see rakenduse eri osade sõltumatut testimist. Rakendus on jaotatud kolmeks suuremaks loogiliseks kihiks: andmepöörduskiht ehk DAL (Data Access Layer), äriloogikakiht ehk BLL (Business Logic Layer) ja rakenduskiht mille struktuur on välja toodud joonisel 12.



Joonis 12. Rakenduse üldstruktuur.

Järgnevalt on kirjeldatud tähtsamaid loogilisi kihte ja nende struktuuri.

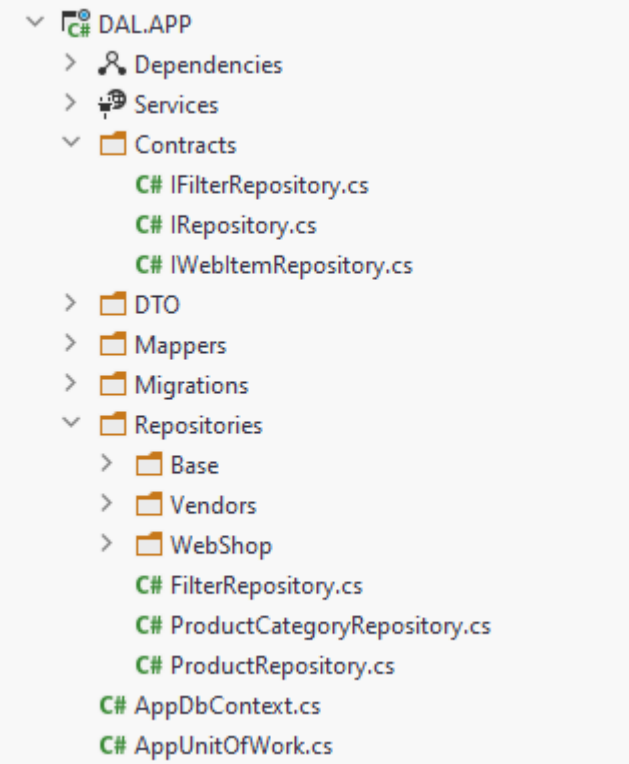
Domain – olemi-suhte diagrammis kirjeldatud olemid ja nendele vastavad objektid koos omavaheliste seostega. Entity Framework raamistiku abil luuakse vastavate klasside põhjal andmebaasi struktuur, mida on näidatud joonisel 13.



Joonis 13. Domeenimudel.

Andmetele juurdepääsu kiht (DAL – Data Access Layer), mille struktuur on kujutatud joonisel 14:

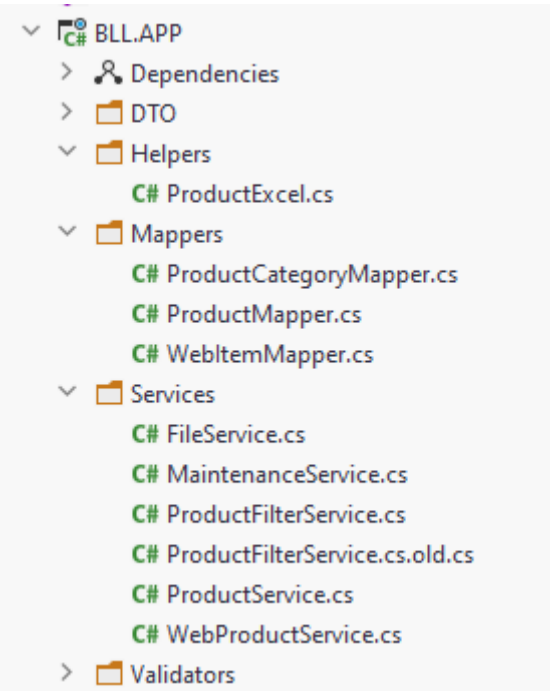
- Contracts.DAL.APP – hoidlate (repository) liidesed, mille kaudu domeeni objekte kasutades käib suhtlus andmebaasiga.
- DAL.DTO – andmeedastusobjektid (DTO – data transfer object) andmetele ligipääsu kihi info edastamiseks.
- DAL.App – andmetele juurdepääsu kihi domeenimudelipõhine funktsionaalsus.



Joonis 14. Andmetele juurdepääsu kiht.

Äriloogika kiht (BLL – Business Logic Layer), mille struktuur on kujutatud joonisel 15:

- BLL.DTO – äriloogika kihi andmeedastusobjektid.
- BLL.Base – äriloogika kihi baasfunktsionaalsus.
- BLL.App – äriloogika rakendamiseks liideste põhjal realiseeritud teenused.



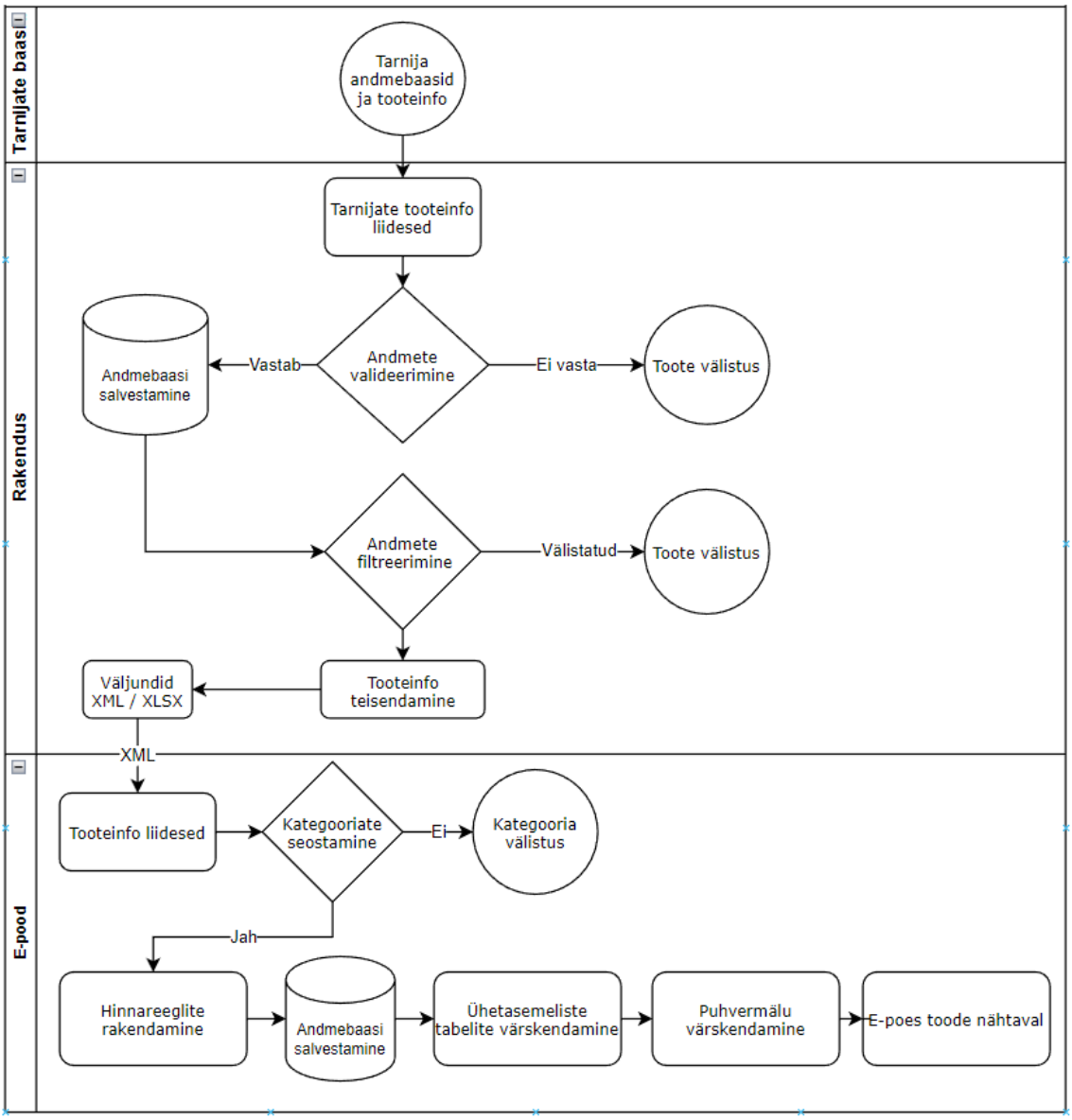
Joonis 15. Äriloogika kiht.

5.3 Töö tulemused

Valminud rakendusele esitatud nõuded said kõik täidetud. Rakendus võimaldab tarnijate tooteinfo allalaadimist, filtreerimist, teisendamist ja selle XML vormingus salvestamist, mida veebipood oskab sisse lugeda. Lisaks loodi ka xslx vormingus väljund, et kasutajad saaksid tooteinfot näha tabeli kujul.

Rakendus paigaldati töötama automaatselt. Selle tarbeks on loodud serverisse ajastatud töö, mis igal ööl kindlal kellaajal käivitab rakenduse ja uuendab tooteinfo. Tihedamalt uuendamine ei ole praktiline, kuna valdav enamus tarnijaid uuendab enda infot vaid üks kord ööpäevas, sagedasem andmete päring ei too seetõttu kaasa info paremat asjakohasust.

Joonisel 16 on kujutatud terve protsess, kuidas tooteinfo liigub ja kus paikneb käesolevas töös kirjeldatud rakendus.



Joonis 16. Tooteinfo liikumise skeem läbi rakenduste.

Tooteinfo laetakse alla tarnija andmebaasist, mis esmalt valideeritakse, et oleks olemas kõik nõutud baasinfo väljad. Kui toode läbib valideerimise, siis salvestatakse see toode andmebaasi, kui ei, siis antud toode välistatakse. Järgmise etapina küsitakse andmebaasist tooted ja nad läbivad filtreerimiskihi, kus toimub filtreerimine vastavalt äriloogikale . Järgnevalt läbib tooteinfo teisendamise, vastavalt nõuetele tooteinfo vajalikud väljad teisendatakse. Seda protsessi on kirjeldatud põhjalikult peatükis 4.4. Viimase etapina väljastatakse rakenduses loodud tooteinfo XML formaadis FTP serveri kausta, mille kaudu see saab veebipoele kättesaadavaks. Lisaks on eraldi väljund xlsx formaadis, et kasutajad saaksid näha kogu tootevalikut ka tabeli kujul.

Kui veebipood on XML faili sisse lugenud, siis esimeses etapis veebipood seostab tooted kategooriatega kasutades vastavustabelit. Kui toote kategooria on veebipoodi seotud, siis antud tootele määratakse juurdehindlus ja lisatakse toode veebipoe andmebaasi. Selleks, et toode ilmuks veebipoes klientidele nähtavaks peab toode veel läbi veebipoe indekseerimislahenduse paigutatama ühetasemelisse tabelisse ja lisatama puhvermällu.

5.4 Rakenduse suurimad väljakutsed

Käesolevas peatükis kirjeldab autor erinevaid väljakutseid, mis tekkisid töö käigus ja millised lahendused nendele leiti. Tegemist on probleemidega, mida on keeruline ette näha töö algfaasis, küll aga nõuavad need rakenduse nõuete lisamist või muutmist töö käigus.

5.4.1 Pildid

Toodete puhul on kaasas alati ka vähemalt üks link tootepildile. Kuna seda infot suudab URLi olemasolul veebipood ise alla laadida, siis eraldi piltidega tegelema ei pidanud. Kuid ühe tarnija puhul tekkis probleem, et URL küll oli, kuid pilt veebipoodi ei ilmunud. Nüüd andmeid kontrollides selgus, et kuigi URL oli korrektne, siis pilti seal tegelikult ei eksisteerinud. Suureks väljakutseks oli see, et pildi link oli korrektne, server vastas URLi päringule pildiga, millel oli piisav maht, et võiks eeldada, et pilt on olemas. Küll aga oli pilt ise katkine. Selline olukord välistas võimaluse kontrollida serveri vastuse järgi pildi olemasolu, lisaks ei olnud võimalik ka mahu järgi öelda, kas pilt tegelikult on olemas või mitte.

Lahendusena leidis autor Magic.NET [53] mooduli, mis võimaldab .NET raamistikus pilte töödelda. Iga antud tarnija pilt laeti alla ja üritati antud mooduliga avada. Kui moodul viskas erindi, siis võis eeldada, et pildifail on katki. Kui erindit ei tekkinud, siis pilt oli korrektne ja seda võis veebipoes kasutada. Antud lahenduse negatiivne aspekt on see, et iga pilt tuli alla laadida ja avada. See on küll automaatne, kuid muudab rakenduse töö tunduvalt aeglasemaks.

5.4.2 Tootjakood ja EAN ei ole alati unikaalne

EAN (European Article Number) on rahvusvaheline kaubakoodide süsteem, mida kasutatakse toodete identifitseerimiseks ja nende andmete haldamiseks tarneahelas [54]. EAN kood peaks olema igal tootel alati unikaalne [54]. Küll aga selgus, et leidub ka selliseid tootjaid, kes standardit ei järgi ja seetõttu kasutavad nad EAN koode oma toodetel, mis ametlikult on reserveeritud teisele tootjale. Sarnaselt EANile, on probleem ka tootjakoodide puhul. Need on küll iga tootja enda luua, kuid tekib täpselt sama olukord, kus tootekoodi ei saa lugeda alati unikaalseks. Kui esimeses iteratsioonis valiti unikaalseks tunnuseks tootekood, siis peale rakenduse valmimist loeti unikaalsust tähistavaks tunnuseks kombinatsioon tarnijast, brändist ja tootekoodist. EAN probleemile hetkel tegelikult lahendust ei ole, vaid selliseid koode välistatakse. See toob kaasa olukorra, kus antud toode pole nähtav ja kliendid ei saa seda osta.

5.4.3 Andmete kvaliteet

Üks suur väljakutse millega autor kokku puutus oli sisendandmete kvaliteet.

Näitena võib tuua ühe tarnija laoseisu info. Laoseis peaks olema eelduslikult numbriline väärtus. Antud tarnija puhul oli laoseis 0 – 50 koguseni numbriline, kui laoseis ületas 50 ühikut, siis kuvati laoseisu väljal info „Rohkem kui 50“. Antud probleemi lahenduseks tuli hakata kontrollima ka laoseisu välja sõnalisi väärtusi ja muuta need numbriliseks, kui võimalik. Kui see ei õnnestunud, siis märgiti toote laoseisuks väärtus 0.

Teise näitena võib tuua ühe üldisema probleemi, nimelt tootekirjelduste koostamise. Tootekirjeldus on tekst, mis sisaldab infot toote kohta. Probleem tekib sellest, kus ja kuidas seda tootekirjeldust hoitakse. Vaba tekstina hoitud info pole liigendatud ega vormindatud, ja ning kliendil on seda väga ebamugav lugeda. Kui aga tekst eelnevalt ära vormindada, siis tihti vormindatakse see mingi rakenduse järgi, mis omakorda tähendab et teistesse rakendustesse see ei pruugi sobida. Sellele probleemile tegelikult head lahendust ei olegi. Alternatiivina võib kasutada kolmandate osapoolte tooteinfo süsteeme, kus tooteinfo on ühtlustatud kujul olemas (seda ka ettevõttes kasutatakse). Need on paraku aga tasulised ja info kvaliteeti võib lugeda rahuldavaks, kuid kindlasti mitte väga heaks. Sellisteks näideteks on 1WorldSync (endise nimega CNet) [55] ja ettevõttes kasutusel olev Icecat [48].

5.4.4 Vajalikku tooteinfot ei ole liideses olemas.

Tarnijal on olemas küll liides, kust küsida tooteinfot, kuid antud liideses ei kajastata kõike vajaminevat baasinfot. Konkreetse näitena saab tuua, et puudu oli tootepilt ja kirjelduse väli. Küll on see info olemas tarnija veebilehel. Antud probleemi lahenduseks loodi lisamoodul, mis võttis liidese tooteinfost toote EAN koodi ja selle järgi läks tarnija veebilehele, sisestas selle otsinguväljale, otsingutulemustest avas tootekaardi ja salvestas ära vajaliku info. Lahendus loodi Selenium WebDriverit [56] kasutades, mis on mõeldud põhiliselt küll automaattestide loomiseks, kuid sobis ka antud probleemi lahenduseks. Kindlasti ei ole tegemist korrektse lahendusega, kuid kuna tegemist oli ettevõttele strateegilise partneriga, siis leiti, et tähtsam on leida lahendus, kui oodata, millal tarnija oma tooteinfot täiendab.

5.5 Mõjud ettevõttele

Käesolevas peatükis kirjeldab autor kuidas loodud rakendus mõjutas ettevõtet.

5.5.1 Ajaline võit

Ajalist võitu on antud rakenduse puhul väga keeruline hinnata, kuna käsitsi tehes oleks ajakulu liiga ebamõistlik. Näiteks kui toote lisamine veebipoodi tehakse käsitsi, kulub ühe toote lisamiseks hinnanguliselt umbes neli minutit. Veebipoes on hetkel tooteid kümnetes tuhandetes, nende laoseisud ja hinnad muutuvad igapäevaselt, seetõttu ei ole manuaalselt sellise töö tegemine võimalik. Küll aga oleks võimalik teha sama tööd poolmanuaalselt, ehk laadida tooteinfo alla, muuta see ära näiteks tabeltöötlusprogrammis ja kasutada mõnda import-eksport moodulit (mida on kirjeldatud ka peatükis 3.4). Hinnanguliselt võtaks selline tegevus aega protsesside paika saamisel ca 30 minutit päevas, kuid ka see ei oleks mõistlik ajakulu.

5.5.2 Rahaline võit

Kindlasti tuleb arvestada majandusliku kasu arvutamisel sellega, et antud rakendus on vaid tooteinfo vahendaja rollis, ehk rakendus ilma õigete tarnijate ja tootegruppide valikuta ise majanduslikku kasu ei too. Rakenduse eesmärk on võimaldada tooteid lisada veebipoodi, kuid otsus mis tooted, millise hinnastusega on juba tootejuhtide töö.

Hetkelist lahenduse majanduslikku kasu ettevõttes analüüsides saab öelda, et käive on samal tasemel kui ettevõtte väikseima jaekaupluse käive ning on kasvutrendis. Aprill 2023 seisuga saab välja tuua, et veebipoes olevatest toodetest on ca 1/3 juba läbi liidese tulevad tooted, 2/3 on majandustarkvara kaudu tulev kaubavalik.

5.5.3 Nähtavuse võit

Kaudne kasutegur on ka veebipoe otsingumootoritest leitavuse ja nähtavuse kasv. Mida rohkem on veebipoes tooteid, seda suurem tõenäosus on, et klient tuleb läbi otsingutulemuste veebipoodi. Isegi kui konkreetne toode talle ei ole sobi, suurendab see tõenäosust, et ta vaatab veebipoest alternatiivseid valikuid.

5.6 Edasiarenduste võimalused

Käesolevas peatükis kirjeldab autor võimalusi töö edasiarendusteks. Kirjeldatud edasiarendused on ka planeeritud ellu viia järgmiste arendusetappide raames.

5.6.1 Asünkroonne rakenduse toimimine

Hetkel töötab rakendus järjestikuliselt, ehk programm käivitub ja tarnijate info uuendatakse tarnija kaupa. Uuendamine on ajakulukas, aprill 2023 seisuga on liidestatud 14 tarnijat ja rakenduse tööaeg on umbes 4 tundi. Hetkel see veel probleemi ei valmista, kuna andmeid uuendatakse üks kord öö jooksul, kuid tarnijate lisandumisel võib tekkida olukord, kus ei jõuta ööga valmis. Lahendusena tuleks rakendus ümber ehitada asünkroonseks, ehk erinevate tarnijate tooteinfo uuendataks samaaegselt.

5.6.2 Integratsioon tooteinfo haldussüsteemiga

Rakendusele ei loodud kasutajaliidest, kasutajatele mõeldud väljundiks on Exceli tabel , filtrite loomine käib läbi json failide, toodetele lisainfo lisamine toimub läbi veebipoe. Oleks vaja lisada tooteinfo haldussüsteem, kus tooteinfo lisatakse toodetele juba enne veebipoodi jõudmist. See võimaldab sama infot kasutada tulevikus ka muudeks väljunditeks, näiteks klientidele pakkumiste koostamise rakendus või meie tooteinfo jagamine edasimüüjatele.

5.6.3 Integratsioon majandustarkvaraga

Hetkel toimub toodetele juurdehindluste määramine läbi veebipoe. See tähendab, et kui sama kategooria on olemas ka majandustarkvaras, siis tuleb juurdehindlus määrata ära nii

majandustarkvaras meie sortimendile, kui ka igale tarnijale veebipoes. See on manuaalne ja aeganõudev töö, kus on ka risk eksimusteks. Kui muudetakse mingi kauba kategooria juurdehindlust, siis tuleb seda mitmes kohas teha. Eesmärk on siduda tarnijate kategooriad ettevõttes asuvate tootekategooriatega, et oleks võimalik kasutada majandustarkvara hinnakujunduse reegleid. Kui muudetakse kategooria hinnareegel majandustarkvaras, siis muutub see ka kõikides seotud tarnijate kategooriates.

6 Kokkuvõte

Antud bakalaureusetöö eesmärk oli luua rakendus tarnijate tooteinfo haldamiseks, mis võimaldaks tarnijate tooteinfot alla laadida, seda filtreerida ja teisendada ning veebipoodi edastada. Eesmärgi saavutamiseks koguti nõuded, analüüsiti olemasolevaid lahendusi ja valiti välja sobivad töövahendid rakenduse loomiseks.

Rakendus majutuseks valis autor sisevõrgus asuva Windowsi serveri, kuna see oli ettevõttes juba olemas ja ei vajanud eraldi kulutusi. Analüüsi tulemusena valis autor lahenduse loomiseks C# keele .NET raamistikus, MySQL andmebaasi ja rakendus kirjutati kasutades Jetbrains Rideri integreeritud arenduskeskkonda.

Rakenduse loomine toimus iteratsioonide kaupa, kus esimeses iteratsioonis analüüsiti tarnijate tooteinfo sisendeid ja selle järgi loodi andmemudel. Teises iteratsioonis loodi rakendus, mis võimaldab tooteinfot alla laadida ja seda andmebaasi salvestada. Kolmandas iteratsioonis lisati rakendusele äriloogika kiht, mis võimaldab tooteinfot filtreerida ja nõutavale kujule teisendada. Viimases iteratsioonis loodi rakendusele XML ja XLSX väljundid veebipoe ja kasutajate tarbeks.

Loodud rakendus on ettevõttes kasutusel ja täidab seatud ärilisi eesmärke. Lahendus on ka ettevõttes regulaarselt edasi arendatav, lisatakse uusi tarnijaid ja vastavalt vajadusele ka uut funktsionaalsust.

Antud lahendust saab minimaalsete muudatustega kasutada ka teistes sarnaste vajadustega ettevõtetes, see nõuab rakenduse väljundite kohandamist vastavalt konkreetsetele nõuetele.

Kasutatud kirjandus

[1] „Eesti e-kaubanduse statistika ja trendid“. https://www.e-kaubanduseliit.ee//liidust/statistika (vaadatud 14. aprill 2023).
[2] „Firmast“. https://www.byroomaailm.ee/firmast (vaadatud 14. aprill 2023).
[3] „The Top Five Go-To Requirements Elicitation Methods“. https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/2483/The-Top-Five-Go-To-Requirements-Elicitation-Methods.aspx (vaadatud 14. aprill 2023).
[4] „Functional vs Non-Functional Requirements [Updated 2021]“. https://enkonix.com/blog/functional-requirements-vs-non-functional (vaadatud 24. aprill 2023).
[5] „5 olulist küsimust, mida enne ettevõtte protsesside digitaliseerimist küsida“, *Thorgate Blog*, 8. juuli 2021. http://blog.thorgate.eu/blog/5-olulist-kusimust-mida-enne-ettevotte-protsesside-digitaliseerimist-kusida/ (vaadatud 14. aprill 2023).
[6] „Magento 2 - Mass Product Update and Import - Extension for Magento“. https://www.wyomind.com/magento2/mass-product-update-import-magento.html (vaadatud 14. aprill 2023).
[7] „Import Products for Magento 2“. https://amasty.com/import-products-for-magento-2.html (vaadatud 14. aprill 2023).
[8] „Pimcore - The Future of Data & Experience Management“. https://pimcore.com/en (vaadatud 14. aprill 2023).
[9] „Agile vs. Waterfall Methodology – Forbes Advisor“. https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology/ (vaadatud 14. aprill 2023).
[10] „Choosing the Right Tech Stack for Your Project“, *Velvetech*, 19. september 2019. https://www.velvetech.com/blog/choosing-project-tech-stack-basic-principles/ (vaadatud 14. aprill 2023).
[11] Symfony, „Symfony, High Performance PHP Framework for Web Development“. https://symfony.com/ten-criteria (vaadatud 15. aprill 2023).
[12] „13 Best hosted version control services as of 2023 - Slant“. https://www.slant.co/topics/153/~best-hosted-version-control-services (vaadatud 20. aprill 2023).
[13] „Build software better, together“, *GitHub*. https://github.com (vaadatud 20. aprill 2023).
[14] „Pricing · Plans for every developer“, *GitHub*. https://github.com/pricing (vaadatud 22. aprill 2023).
[15] „Bitbucket | Git solution for teams using Jira“. https://bitbucket.org/product (vaadatud 20. aprill 2023).
[16] Atlassian, „Bitbucket - Pricing“, *Atlassian*. https://bitbucket.org/product/pricing (vaadatud 20. aprill 2023).
[17] Atlassian, „Jira | Issue & Project Tracking Software“, *Atlassian*. https://www.atlassian.com/software/jira (vaadatud 20. aprill 2023).
[18] „Pricing | GitLab“. https://about.gitlab.com/pricing/ (vaadatud 20. aprill 2023).
[19] „Unify the DevSecOps lifecycle with GitLab“. https://about.gitlab.com/stages-devops-lifecycle (vaadatud 20. aprill 2023).

[20] A. Panwar, „Types of Database Management Systems“. https://www.c-sharpcorner.com/uploadfile/65fc13/types-of-database-management-systems/ (vaadatud 24. aprill 2023).
[21] „MariaDB Vs MongoDB Comparison“, *MongoDB*. https://www.mongodb.com/compare/mariadb-vs-mongodb (vaadatud 16. aprill 2023).
[22] „SQL vs NoSQL: Which one is better to use?“, *GeeksforGeeks*, 16. mai 2019. https://www.geeksforgeeks.org/sql-vs-nosql-which-one-is-better-to-use/ (vaadatud 16. aprill 2023).
[23] A. Panwar, „Types of Database Management Systems“. https://www.c-sharpcorner.com/uploadfile/65fc13/types-of-database-management-systems/ (vaadatud 16. aprill 2023).
[24] K. T, „SQL vs. NoSQL: full comparison of features, differences, and more“, *TestGorilla*, 11. november 2022. https://www.testgorilla.com/blog/sql-vs-nosql/ (vaadatud 24. aprill 2023).
[25] „DB-Engines Ranking“, *DB-Engines*. https://db-engines.com/en/ranking/relational+dbms (vaadatud 16. aprill 2023).
[26] „SQLite Documentation“. https://www.sqlite.org/docs.html (vaadatud 16. aprill 2023).
[27] „PostgreSQL: Documentation“. https://www.postgresql.org/docs/ (vaadatud 16. aprill 2023).
[28] „MySQL :: MySQL Documentation“. https://dev.mysql.com/doc/ (vaadatud 16. aprill 2023).
[29] MashaMSFT, „SQL Server technical documentation - SQL Server“. https://learn.microsoft.com/en-us/sql/sql-server/ (vaadatud 16. aprill 2023).
[30] „Database Documentation“, *Oracle Help Center*. https://docs.oracle.com/en/database/ (vaadatud 16. aprill 2023).
[31] „Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others“, *AltexSoft*. https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/ (vaadatud 16. aprill 2023).
[32] „Microsoft SQL Server Express considerations | Deep Security“. https://help.deepsecurity.trendmicro.com/11_0/aws/Get-Started/sql-express.html (vaadatud 15. aprill 2023).
[33] „SOLID: The First 5 Principles of Object Oriented Design | DigitalOcean“. https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design (vaadatud 23. aprill 2023).
[34] „Stack Overflow Developer Survey 2022“, *Stack Overflow*. https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022 (vaadatud 15. aprill 2023).
[35] „Web programming languages: the best languages for web development“, *IONOS Digital Guide*, 11. märts 2019. https://www.ionos.com/digitalguide/websites/web-development/web-programming-languages/ (vaadatud 16. aprill 2023).
[36] „Top 20 Best Programming Languages To Learn in 2023“, *Simplilearn.com*, 18. mai 2015. https://www.simplilearn.com/best-programming-languages-start-learning-today-article (vaadatud 15. aprill 2023).

[37] M. Glynska, „Laravel vs. NET - Which Framework Will Fit Your Future Project“, *A-Team Global*, 11. november 2021. https://a-team.global/blog/laravel-vs-net-which-framework-will-fit-your-future-project/ (vaadatud 22. aprill 2023).
[38] „C# Programming - The State of Developer Ecosystem in 2022 Infographic“, *JetBrains: Developer Tools for Professionals and Teams*. https://www.jetbrains.com/lp/devecosystem-2022 (vaadatud 22. aprill 2023).
[39] „Laravel - The PHP Framework For Web Artisans“. https://laravel.com/ (vaadatud 24. aprill 2023).
[40] „Laravel Zero“, *Laravel Zero*. https://laravel-zero.com (vaadatud 22. aprill 2023).
[41] „ASP.NET vs .NET | Top 3 Excellent Comparison to Know in Detail“, *EDUCBA*, 11. september 2018. https://www.educba.com/asp-dot-net-vs-dot-net/ (vaadatud 24. aprill 2023).
[42] „Slant - 8 Best C# IDEs as of 2023“, *Slant*, 1. aprill 2023. https://www.slant.co/topics/4118/~c-ides (vaadatud 16. aprill 2023).
[43] „Features - Rider“, *JetBrains*. https://www.jetbrains.com/rider/features/ (vaadatud 23. aprill 2023).
[44] „Visual Studio: IDE and Code Editor for Software Developers and Teams“, *Visual Studio*. https://visualstudio.microsoft.com (vaadatud 23. aprill 2023).
[45] „Visual Studio Code - Code Editing. Redefined“. https://code.visualstudio.com/ (vaadatud 23. aprill 2023).
[46] „What is Postman? Postman API Platform“, *Postman API Platform*. https://www.postman.com/product/what-is-postman/ (vaadatud 25. aprill 2023).
[47] „Postman - API Testing Tool: What It is, Tutorial - Javatpoint“, *www.javatpoint.com*. https://www.javatpoint.com/postman (vaadatud 25. aprill 2023).
[48] „What’s the EAN code?“ https://www.minderest.com/blog/what-is-ean-code (vaadatud 17. aprill 2023).
[49] „Icecat: all-in-one platform for content and syndication“. https://icecat.com/ (vaadatud 17. aprill 2023).
[50] „QSEE SuperLite. Get the software safely and easily.“, *Software Informer*, 8. aprill 2023. https://qsee-superlite.software.informer.com/ (vaadatud 25. aprill 2023).
[51] IEvangelist, „Testing in .NET - .NET“, 23. märts 2023. https://learn.microsoft.com/en-us/dotnet/core/testing/ (vaadatud 24. aprill 2023).
[52] „NUnit vs. XUnit vs. MSTest: Comparing Unit Testing Frameworks In C#“, *LambdaTest*, 22. märts 2021. https://www.lambdatest.com/blog/nunit-vs-xunit-vs-mstest/ (vaadatud 22. aprill 2023).
[53] D. Lemstra, „The .NET library for ImageMagick: Magick.NET“. 16. aprill 2023. Vaadatud: 17. aprill 2023. [Online]. Available at: https://github.com/dlemstra/Magick.NET
[54] „Content Solutions - 1WorldSync“. https://contentsolutions.1worldsync.com/en/ (vaadatud 18. aprill 2023).
[55] „Selenium“, *Selenium*. https://www.selenium.dev/ (vaadatud 22. aprill 2023).

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

Mina, Sebastian Sõeruer

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Tarnijate tooteinfo haldamise rakendus veebipoele Büroomaailma näitel“, mille juhendaja on Kristiina Hakk.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

24.04.2023

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