TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jan-Erik Kalmus IADB186008

Konkurentsiameti soojusvaldkonna põhjendatud hinna kooskõlastamise tarkvara prototüüp

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad
Andreas Türk
Bakalaureusekraad

Tallinn 2021

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: Jan-Erik Kalmus

17.05.2021

Annotatsioon

Bakalaureusetöö eesmärk on luua Konkurentsiametile prototüüp tarkvarast, mis kasutab sisendina soojusettevõtja majandusaasta andmetest koosnevat taotlust ning annab taotlust menetlevale analüütikule väljundi, millest on võimalik soojuse põhjendatud hinna kooskõlastamisel lähtuda.

Töö käigus kavandatakse ja arendatakse veebirakendus, mis demonstreerib lähteülesande lahendatavust. Autor põhjendab arenduses kasutatud tehnoloogiate valikuid ning tutvustab ülesande lahendamisel kasutatud metoodikaid.

Rakenduse nõuete määramisel lähtuti minimaalsest funktsionaalsusest, millega on võimalik kinnitada, et Konkurentsiameti visioon on teostatav. Lähteülesande edukal lahendamisel on ette nähtud rakenduse arendamise jätkamine.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 7 peatükki, 11 joonist, 2 tabelit.

Abstract

Prototype for Determining Justified Price of District Heating for the Competition Authority of Estonia

The purpose of this thesis is to create prototype software for the Competition Authority of Estonia. Price applications for district heating, filed by heat distributors or manufacturers, will be used as input. The prototype is to provide an output, based on which the Competition Authorities analyst could successfully determine whether the proposed heating price is justified or requires further investigation instead.

The thesis will cover the design and development of a web application, the purpose of which is to act as proof of concept. Technologies and methods used in the application are introduced and thoroughly explained by the author.

Requirements for the web application were determined by the minimal amount functionality required to demonstrate the viability of the solution. If a viable solution for the problem is found, the development of the application is to be continued in the future.

The thesis is in Estonian and contains 34 pages of text, 7 chapters, 11 figures, 2 tables.

Lühendite ja mõistete sõnastik

analüütik	menetluse käigu eest vastutav ametiisik
boilerplate	lähtekood, mida on kirjutatud rakenduse arendamise jaoks vajaliku baasfunktsionaalsuse loomiseks
commit	lähtekoodi muudatusi hoidlasse salvestav käsklus
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
extension method	klassi instantsile funktsionaalsust lisav meetod, mis on loodud staatilise meetodina väljaspool klassi deklaratsiooni
funktsionaalsed nõuded	nõuded infosüsteemile, mis määravad infosüsteemi omaniku vajadused sisendinfo muutmiseks väljundinfoks
HTML	*Hypertext Markup Language*
kratt	tehisintellekti süsteem, mis põhineb õppimisvõimelisel algoritmil ning täidab traditsiooniliselt inimese tehtavaid toiminguid
LINQ	*Language Integrated Query* - C# programmeerimiskeelde integreeritud süntaks, mis võimaldab erinevatel andmekogumitel põhinevaid päringuid
mittefunktsionaalsed nõuded	nõuded infosüsteemile, mis keskenduvad süsteemi käitumise täpsustamisele, mitte konkreetsete funktsionaalsuste või omaduste kirjeldamisele
MVC	*Model View Controller*
ORM	*Object relational mapping*
partial view	osaline vaade. ASP.NET MVC raamistiku vaadetes kasutatav alamkomponent, mis võimaldab elementide taaskasutatavust ja modulaarsust
regulatsiooniperiood	12-kuuline periood, mille põhjendatud hinnakomponendid on aluseks soojuse piirhinna arvutamisel. Regulatsiooniperiood ei pea kattuma kalendriaasta ega ettevõtte majandusaastaga
trassikadu	tootmisüksuse tootmis- ja müügimahu vahe

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Konkurentsiameti üheks ülesandeks on igapäevaelus oluliste teenuste, nagu näiteks kaugkütte (soojuse) ning vee- ja kanalisatsiooniteenuste hindade kooskõlastamine. Reguleerimise eesmärk on hoida nende tähtsate teenuste hinnatase nii teenusepakkuja kui tarbija jaoks põhjendatud tasemel.

Kui mõnel sellist teenust pakkuval ettevõttel tekib vajadus teenuse hinda muuta, on selleks vajalik esitada Konkurentsiametile taotlus koos planeeritava hinnamuutuse vajalikkust ja põhjust toetavate andmetega. Praeguses praktikas on ettevõtete esitatud taotluste ja sellega seotud andmete analüüs inimese ülesandeks.

Väiksema majandusmõjuga ettevõtete esitatavate taotluste analüüsimiseks kuluv aeg on aga märkimisväärselt suur tulenevalt nende arvukusest. Nende taotluste arvukusest tekkinud töökoormuse vähendamisel oleks taotlustega tegelevatel töötajatel võimalik põhjalikumalt süveneda kordades mahukamatesse suurema majandusmõjuga ettevõtete taotlustesse.

Väärtusliku ajalise – ja seeläbi ka rahalise – ressursi säästmiseks on Konkurentsiamet avaldanud soovi tehnilise lahenduse loomiseks, mille eesmärk on hinnataotluste analüüsiga seotud tööprotsesside osaline automatiseerimine ning märgatav lihtsustamine.

Töö tellija esialgne visioon oli luua kratt, mis tuginedes Konkurentsiameti aja jooksul kogutud andmete analüüsile abistab töötajail konkreetse taotlustega seotud andmeid analüüsida ning põhjendatud hinna määramiseks vajalike arvutusi läbi viia.

Antud lõputöö raames autor planeerib, analüüsib ja realiseerib vastavalt Konkurentsiameti välja toodud funktsionaalsetele nõuetele ja eesmärkidele prototüüplahenduse, mille abil on võimalik demonstreerida sõnastatud probleemi lahendatavust. Probleemile töötava lahenduse loomise puhul on ette nähtud järgnevate arendusetappide täitmine ning seejärel töötava lahenduse praktikas rakendamine.

1.1 Metoodika

Käesoleva diplomitöö käigus tutvustatakse esmalt töö aluseks olevat probleemi ning selle lahendamise vajalikkust. Diplomitöö skoop määratakse autori ja Konkurentsiameti vahelise suhtluse järel, lähtudes Konkurentsiameti visioonist ja eesmärkidest.

Diplomitöös tutvustatakse põhjalikult lähteülesande valdkonda ning selles kasutatavat metoodikat, mille tundmine on analüüsi ja arenduse käigus tehtud otsuste mõistmiseks ja põhjendamiseks oluline.

Analüüsi käigus planeeritakse rakenduse arendust ning põhjendatakse arendusega seotud valikuid: rakenduse arhitektuur, kasutatavad tehnoloogiad ja raamistikud, rakendatavad arendusmustrid ning valitud töövahendid. Tutvustatakse ka kavandatud testimise metoodikat.

Analüüsile järgneb töö praktiline osa, kus kavandatud prototüüprakendus realiseeritakse vastavalt eelnevalt sätestatud teoreetilistele lähtepunktidele.

Töö lõpuosas tehakse kokkuvõte saavutatust ning analüüsitakse töö tulemuse vastavust töö alguses sõnastatud nõuetele ja skoobile. Töö vastavust nõuetele hinnatakse lähtudes testimise tulemusest ja tellija tagasisidest. Diplomitöö lõpus kirjeldatakse rakenduse planeeritavat arengukäiku sõltuvalt töö tulemustest.

2 Probleem

Kaugkütte valdkonna hindade kooskõlastamine on vastavalt kaugkütteseadusele Konkurentsiameti ülesandeks [1]. Kui soojusvaldkonnas tegutsev ettevõte soovib soojuse hinda määrata, tuleb see eelnevalt kooskõlastada Konkurentsiametiga [1], [2].

Hinna kooskõlastamine hõlmab vajalike andmete esitamist Konkurentsiametile, kes andmeanalüüsi ja arvutuste tulemusel otsustab, kas ettevõtja poolt taotletav hind ning seda moodustavad tulu- ja kulukomponendid on vajalikud ja põhjendatud tasemel [3]. Andmete esitamisel on ettevõtjal töö lihtsustamiseks soovitav järgida Konkurentsiameti poolt koostatud taotluse formaati [4], [5].

Menetlusprotsessi tulemusena kooskõlastatakse ettevõtjaga soojusele piirihind, mis on tarbijale pakutava hinna maksimaalne piir. Ettevõtja võib kuni kolmeaastaseks perioodiks Konkurentsiametiga kooskõlastada ka soojuse hinnavalemi, kuhu on muutujatena sisse arvestatud ettevõtte tegevusest sõltumatuid kulukomponente, nagu näiteks soojuse tootmiseks kasutatava kütuse hind [6].

2.1 Olemasolev lahendus

Senises praktikas on taotluse menetlusprotsessis põhilise vahendina kasutatud Microsoft Excelit, mis on tänapäeval üks tuntuimad andmete analüüsiks ja visualiseerimiseks mõeldud tabeltöötlusprogramme [7].

Menetlemise käigus kasutab analüütik põhiliselt kahte erinevat Exceli faili: esimesena ettevõtte poolt esitatud taotlust ning teisena võrdlusanalüüsi faili, milles asuvad eelmiste taotluste ajaloolised andmed erinevate näitajate analüüsiks olulised lisatabelid.

Taotlus sisaldab ettevõtte poolt esitatud majandusandmeid, mis annavad põhjaliku ülevaate ettevõtte kuludest ja tuludest ning regulatsiooniperioodi prognoosidest.

Võrdlusanalüüsi fail koosneb põhiliselt varasemalt taotluse esitanud ettevõtete üldandmetest ja nende hinnakomponentide arvulistest väärtustest, mille alusel arvutatakse võrdlusanalüüsi jaoks iga komponendi kaalutud keskmine.

Igal menetlemisega tegeleval analüütikul on tööprotsessi läbiviimiseks välja kujunenud individuaalne metoodika. Näiteks tõstetakse olemasolevad andmed erinevateks kontrollideks eraldi faili või sooritatakse kontrollarvutused taotluse sees andmete kõrval. Vaatamata välja kujunenud meetoditele sooritab iga analüütik kõik hinnataotluse menetlemisega seotud arvutused ja kontrollid käsitsi.

2.2 Probleemi kirjeldus

Valdkondade üleselt on Konkurentsiameti andmetel ligikaudu 300 reguleeritavat ettevõtet, kelle summeritud käive on 700 miljonit eurot. Nendest 300-st ettevõttest 10% moodustavad käibest 95%, mistõttu kulub ebaproportsionaalselt palju ressurssi väikse majandusliku mõjuga 270 väikeettevõtja andmete analüüsimiseks.

Vaatamata Exceli arvukatele võimalustele on praegune lähenemine äärmiselt ebapraktiline, sest taotlusega tegelev töötaja peab andmeid kombineerima – sellest tulenevalt neid ka otsima - erinevatest tabelist või failidest. Lisaks kaasneb andmete käsitsi sisestamisel teatud veaprotsent, mis võib statistilise analüüsi puhul tulemusi märkimisväärselt mõjutada [8].

Taotluste andmete valideerimine on samuti analüütiku vastutada. Kui taotluse andmetest leitakse vigu või ebakõlasid, teavitatakse ettevõtet puudustest ning taotlust ei võeta menetlusse. Taotluse andmestiku algne valideerimine kujutab aga endast ajalist lisakulu analüütiku jaoks.

Taotluste andmed peavad olema põhjalikud ning toetama ettevõtte enda järeldusi, mistõttu on esitatavate andmete maht ka väiksemate ettevõtete puhul suur. Ühe taotluse täismenetluse puhul kulub menetlejal süvaanalüüsiks ligikaudselt 6 kuni 7 tundi.

Soojusvaldkonnas tehakse ligikaudu 55 täismenetlust aastas. Sellest tulenevalt oleks lahenduse loomise korral kokku hoitud töötundide arv maksimaalselt 385. Diplomitöö kirjutamise ajal on Eestis keskmine brutotunnipalk 2020. aasta neljanda kvartali andmete põhjal 8,7 eurot tunnis [9], mille puhul oleks aastas rahaline kokkuhoid 3349,5 eurot.

Kokkuvõttes kujutab menetlusprotsess endast märkimisväärset ajakulu ning ebamugavust nii taotluse menetlevale analüütikule kui ka seda esitanud ettevõtjale.

2.3 Planeeritav lahendus

Taotluste menetlemisega seotud ressursikulu vähendamiseks, protsessi mugavdamiseks ja selle läbipaistvuse suurendamiseks on Konkurentsiamet pöördunud Registrite ja Infosüsteemide Keskuse poole sooviga luua ühtne veebipõhine menetluskeskkond.

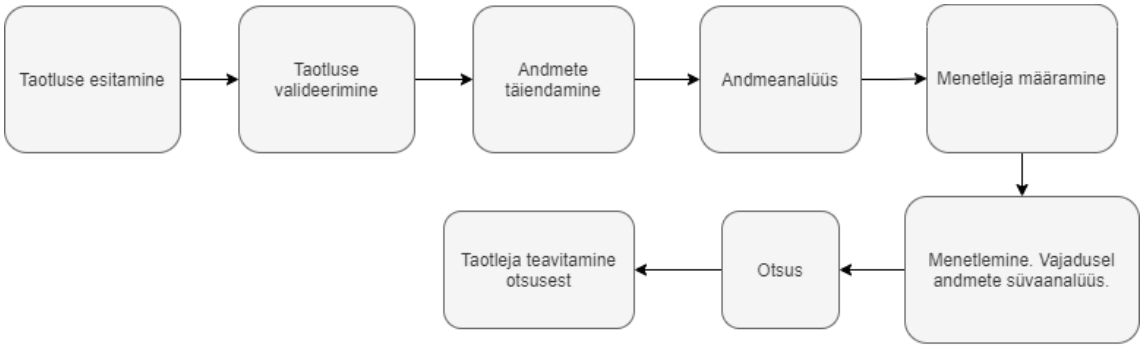
Planeeritava menetluskeskkonna arendus on jagatud kolme suuremasse etappi:

1. Esmase põhjendatud hinna määramise prototüüplahenduse loomine ühe valdkonna näitel, demonstreerimaks menetluskeskkonna võimalikkust.

2. Veebipõhise menetluskeskkonna loomine Konkurentsiameti ja ettevõtja vahelise suhtluse koondamiseks.

3. Menetluskeskkonna laienemine teistele Konkurentsiameti hinnaregulatsiooni alla kuuluvatele valdkondadele.

Üldine visioon näeb ette ühtse menetluskeskkonna loomist, kuhu koondatakse kogu menetlusprotsess tervikuna: ettevõtja ja menetleja vaheline suhtlus, taotluste esitamine, taotluste andmete valideerimine ning taotluste analüüs. Ühtne menetluskeskkond kiirendaks ja lihtsustaks taotlusprotsessi ning samuti suurendaks selle läbipaistvust.



Joonis 1. Menetluskeskkonna tööprotsessi visioon

Käesoleva diplomitöö eesmärk on luua menetluskeskkonna arenduse esimeses etapis kirjeldatud prototüüplahendus, mis võtab sisendiks soojusvaldkonna hinnataotluse ning seejärel andmeanalüüsi ja arvutuste tulemusena pakub menetlejale väljundi, millele tuginedes on menetlejal võimalik taotluse osas otsus langetada või vajadusel taotluse süvaanalüüsi jätkata.

3 Soojuse põhjendatud hinna määramine

Arvestades lähteülesande valdkonna spetsiifilisust, peab autor oluliseks lahendatava probleemiga seotud metoodikaga tutvumist.

3.1 Ülevaade

Lihtsustatult jaguneb soojuse põhjendatud hinna määramise protsess kolmeks etapiks:

- **Taotluse esitamine** – Ettevõtja esitab allkirjastatud soojuse hinnataotluse [3]. Konkurentsiamet kontrollib taotluse nõuetekohasust ning vajadusel nõuab puuduste eemaldamist [3].

- **Menetlemine** – Menetluse käigus kontrollitakse ettevõtja poolt esitatud andmete vastavust äriregistrist saadavate majandusaasta aruannetega [3]. Võrdluse käigus leitud puudujääke palutakse ettevõtjal selgitada [3].

- **Kooskõlastamine** – Kui ettevõtte taotlus on menetlusprotsessi läbinud, alustatakse soojuse piirihinna ja vajadusel hinnavalemi kooskõlastamist [3].

3.2 Taotlus

Reguleeritav ettevõte esitab Konkurentsiametile andmed, millega põhjendatakse nii taotletava soojuse piirihinna kui ka tarbijale rakenduva soojuse hinna kujunemist [1], [6]. Taotlus koosneb ettevõtte viimase kolme majandusaasta tegelikest andmetest ning regulatsiooniperioodi prognoosist [3], [5]. Taotluse peab ettevõtja esitama iga teenindatava võrgupiirkonna kohta eraldi [5].

Taotluse sätestatud struktuur, selle erinevate osade sisulised selgitused ja nõuded on põhjalikult kirjeldatud Konkurentsiameti loodud dokumendis „Hinna- või järelevalvemenetluse läbiviimiseks vajalike andmete esitamise juhend soojusettevõtjatele“ [5].

3.3 Menetlus

Tarbija jaoks soojuse põhjendatud hinna kooskõlastamise metoodika on Konkurentsiameti poolt avalikult kirjeldatud [6]. Soojuse põhjendatud hinna kooskõlastamise metoodikat rakendatakse kõikidele Konkurentsiameti regulatsiooni alla kuuluvatele ettevõtetele võrdsetel põhimõtetel [6].

3.3.1 Hinna komponendid

Soojuse hinna määravad ainult põhjendatud kulud [6]. Hinda kujundavate komponentide põhjendatust ja kuluefektiivsust hinnates lähtub Konkurentsiamet ettevõte esitatud kuludest ja tehnilistest näitajatest [6]. Põhjendatud kulud saab liigitada kolmeks:

- **Muutuvkulud** – Muutuvkulude alla arvestatakse näiteks kulud soojuse tootmiseks vajalikule kütusele, kulud soojuse ostmiseks teistelt tootjatelt, keskkonnatasud ning muud muutuva iseloomuga kulud (elektrienergia, vee- ja kanalisatsiooniteenus, tuha käitlemine jms) [6].

- **Tegevuskulud –** Tegevuskulude alla liigitatakse näiteks seadmete ja rajatiste jooksvad hooldus ja remondikulud, tööjõukulud, bürookulud või müügitegevusega seotud kulud [6].

- **Kapitalikulu –** Kapitalikuluga teenib ettevõte tagasi põhivara soetamiseks tehtud kulutused soetatud põhivara kasuliku tehnilise eluea jooksul [6].

Soojuse hinda ei lülitata kulusid, mis ei ole otseselt seotud soojuse tootmise, jaotamise või müügiga nagu näiteks dividendide kulud, finantskulud, erisoodustuskulud või sponsorlusega seotud kulud [6]. Tarbija ei ole kohustatud tasuma ettevõttele esitatud nõuete ega õigusaktide alusel määratud rahaliste karistuste eest [6].

Komponentide kontrollimisel ja analüüsimisel rakendatakse meetodeid, mis hõlmavad esitatud andmete põhjaliku analüüsi või võrdlust, tuginedes erinevatele andmestikus esile toodud näitajatele, arvutustele, ajaloolistele andmetele või eksperthinnangutele [6].

Kui ettevõte on soojuse hinda lülitanud kulud, mis on vastuolus Konkurentsiameti kuluartiklile arvutatud piirväärtustega või reeglitega, siis on Konkurentsiametil õigus nõuda ettevõttelt täiendavaid põhjendusi [6].

3.3.2 Tehnilised nõuded

Tuginedes Eesti energiatõhususe direktiivi põhieesmärkidele [10], lähtudes kaugkütteseadusest ning arvestades majandus- ja kommunikatsiooniministri 22.06.2011 määruses nr 51 „Soojuse müügi ajutise hinna kehtestamise kord“ toodud tingimusi, soojusettevõtjatele sätestatud nõuded tehnilistele näitajatele motiveerimaks soojusettevõtjaid investeerima energia tõhusamasse tootmisesse ja jaotamisse [6].

3.4 Kooskõlastamine

Kui ettevõtja esitatud andmed on korrektsed ning kõik soojushinna komponendid on adekvaatselt põhjendatud on viimaseks sammuks soojuse piirihinna või selle valemi kooskõlastamine. Ettevõte võib kooskõlastamiseks esitada oma soovitud hinnavalemi, kuid metoodikas on lisaks sätestatud standard hinnavalem (Lisa 2).

Hinnavalemi kooskõlastamise eesmärk on garanteerida ettevõttele võimalus müüa soojust võttes arvesse ettevõtte tegevusest sõltumatuid tegureid, näiteks soojuse tootmiseks kuluva kütuse hinna muutust [6]. Hinnavalemi kooskõlastamist võib ettevõtja taotleda kuni kolmeks aastaks [6].

4 Analüüs

Rakenduse eeldatav funktsionaalsus sõnastati pärast mitut kohtumist tellijaga, kus tulevane kasutaja tutvustas autorile oma tööprotsessi hinnataotluste menetlemisel ja visiooni võimalustest, mida kasutaja rakenduse kasutamisel eeldaks.

Oluline faktor rakenduse skoobi kujunemisel on ajaline ressurss. Loodav lahendus on prototüüp, mille põhieesmärk on demonstreerida probleemi lahendatavust.

Prototüübi loomisel on oluline rakenduse erinevad osad prioriseerida vastavalt tellija soovile, et kasutada ülesande lahendamiseks saadaolevaid ressursse võimalikult efektiivselt. Sellest lähtuvalt tuleb keskenduda tellija jaoks vajaliku äriloogika põhjalikule loomisele ja testimisele, pidades silmas võimaliku edasist arendamist ning prototüübi võimaliku esialgset kasutuselevõttu menetlusprotsessi toetava tööriistana.

Samuti tuleb rakenduse kavandamisel silmas pidada, et sisendandmete allikas ja töödeldud andmete hoiustamise asukoht ja viis võivad tulevikus muutuda. See tähendab, et esialgselt on prototüübi ülesanne kasutada võrdlusandmete allikana senini kasutuses olevat võrdlusanalüüsi faili, kuid prototüübi edasiarendusel peab olema võimalik üleminek relatsioonilisele andmebaasile.

Senises praktikas on Konkurentsiameti tööprotsessides kasutusel vaid Microsoft Exceli failid – sellest tulenevalt muid tabeltöötlusprogrammide faile või formaate prototüübis sisendiks ei kasutata.

4.1 Funktsionaalsed nõuded

Rakenduse esialgseteks põhikasutajateks saavad olema Konkurentsiameti analüütikud. Lähtudes kasutaja vajadustest sõnastas autor järgnevad funktsionaalsed nõuded:

- Kasutaja soovib Excel formaadis soojusvaldkonna taotluseid sisendina kasutada.
- Kasutaja soovib, et hinnataotlust valideeritaks automaatselt.

- Kasutaja soovib, et kõik menetluses vajalikud arvutused sooritataks automaatselt.
- Kasutaja soovib kasutada võrdlusanalüüsi tabeli andmeid.
- Kasutaja soovib täiendada võrdlusanalüüsi tabelit menetluse tulemustega.
- Kasutaja soovib taotluse andmeid võrrelda võrdlusanalüüsi faili andmetega.
- Kasutaja soovib võrdlusanalüüsi failist võrdlemiseks kasutatavat valimit määrata.
- Kasutaja soovib automaatselt leida ebakõlad taotluse andmetes.

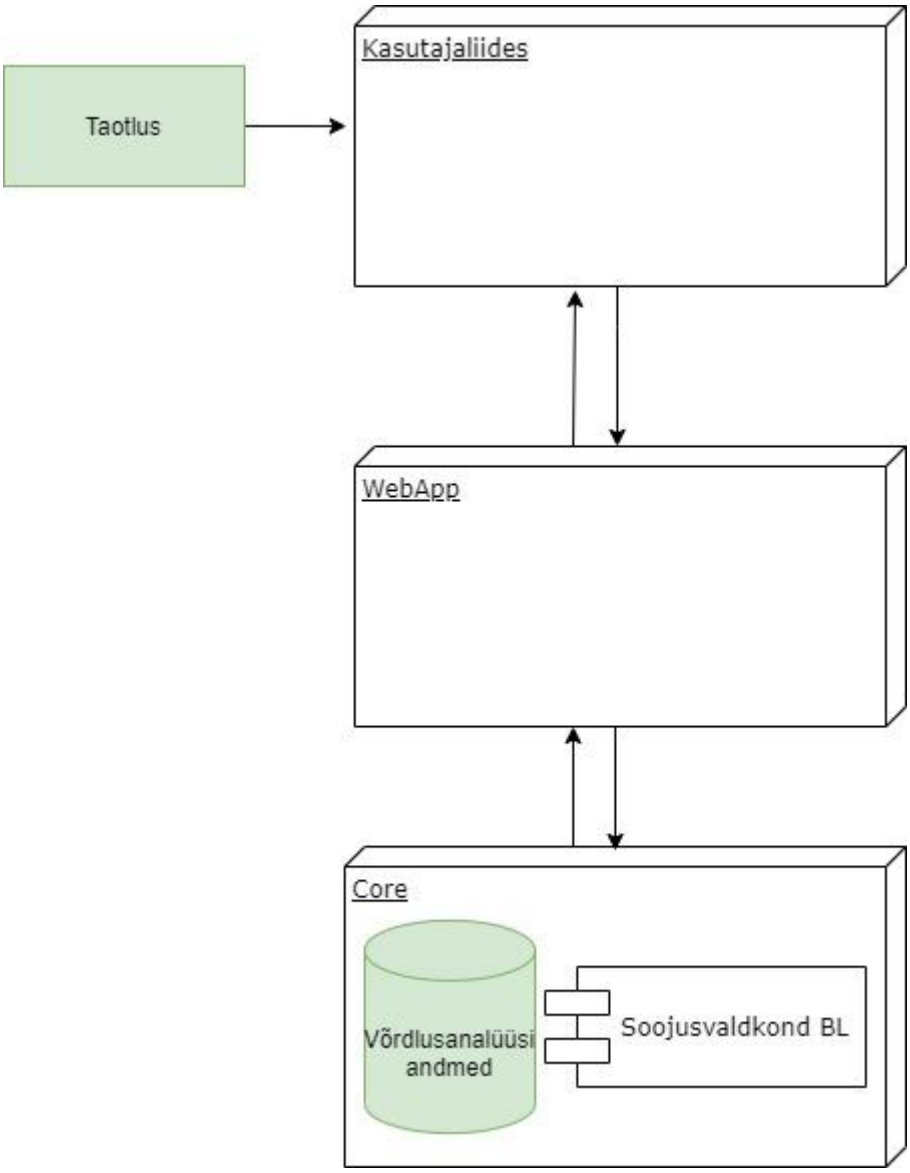
4.2 Mittefunktsionaalsed nõuded

Rakenduse mittefunktsionaalsed nõuded on sõnastatud järgnevalt:

- Kasutaja soovib, et rakendust saaks kasutada senise tööprotsessiga paralleelselt.
- Kasutaja soovib, et kasutajaliides oleks kiirelt õpitav ja lihtsa struktuuriga.
- Kasutaja soovib, et kasutajaliides oleks eesti keeles.
- Kasutaja soovib, et rakendus oleks toetatud kaasaegsetel brauseritel.

4.3 Arhitektuur

Nõuetest tulenevalt on prototüübi põhieesmärk andmetöötlus ning sellele tuginedes väljundi loomine. Prototüüp ei õpi olemasolevatest andmetest ega langeta ühtegi menetlusprotsessiga seotud otsust, vaid automatiseerib andmete valideerimist vastavalt äriloogikale ning võimaldab äriprotsessiga seotud vajalike statistilisi arvutusi. Seetõttu ei ole prototüüpi korrektne nimetada kratiks, nagu tellija analüüsieelne visioon lahendusest ette nägi.



Joonis 2. Rakenduse kavandatav arhitektuur

Prototüüp jaguneb arhitektuuriliselt (Joonis 2) kolmeks osaks:

- Kasutajaliides – võimaldab kasutajal prototüübi põhifookuseks olevat äriloogikat rakendada.
- *WebApp* – võimaldab kasutajaliidesel liidestuda äriloogikaga ning vajalike sisendandmeid töötlemiseks edastada.
- *Core* – sisaldab rakenduse äriloogikat ning selle teostamist võimaldavaid mooduleid ja protsesse.

Lähtudes asjaolust, et esialgselt soovitakse prototüüpi kasutada võrdlusanalüüsi ja taotluse Exceli failide vahelise moodulina, kus andmeid loetakse ühest ja tulemused kirjutatakse ja loetakse teisest, tuleks arhitektuuris kasutada *repository* [11] arendusmustrit. Nimetatud arendusmustri rakendamine võimaldab rakenduse võrdlusanalüüsi andmete allikat tulevikus hõlpsasti vahetada. Rakenduse kavandatav sisemine struktuur on leitav diplomitöö kolmandas lisas.

4.4 Tehnoloogiad

Tehnoloogiate valimisel tuleb lähtuda eelkõige kavandatava rakenduse eesmärgist ja skoobist. Käesoleva diplomitöö tehnoloogiad valitakse kooskõlas järgnevate eesmärkidega:

- Funktsionaalsus – kasutatavad tehnoloogiad võimaldavad piisavalt baasfunktsionaalsusi, et hoida kokku alusfunktsionaalsuste ehk *boilerplate* arenduste arvelt. Tehnoloogia peab oma võimalustega võimalikult palju toetama lähteülesande täitmist.

- Moodulid - tehnoloogia peab toetama hõlpsasti valmiskomponentide integreerimist näiteks paketihalduri abil. Diplomitöö kontekstis on oluline näiteks Microsoft Exceliga failidega töötamiseks mooduli leidmine ning tulevikus võimalikku riikliku andmevahetuskihi X-teega [12] integreerimist silmas pidada.

- Töökindlus ja populaarsus – tehnoloogia on tõestanud ennast tarkvara arendamisel usaldusväärse ja populaarse töövahendina. Tehnoloogia ja selle funktsionaalsus on hästi dokumenteeritud ning tehnoloogia arendaja jätkab tuge või uuendamist kavandatava rakenduse eeldatava elutsükli jooksul.

- Ajaline ressurss – tehnoloogia kasutamine võimaldab autoril hoida kokku aega. Seetõttu on eelistatud tehnoloogiad, millega autor on eelnevalt kokku puutunud või mille õppimine ei nõua liialt palju aega.

4.4.1 Serveripoolne tehnoloogia

Serveripoolse tehnoloogia valimisel võetakse esmalt valiku aluseks populaarseimad, seeläbi ka ennast turul koos raamistikega end kõige rohkem tõestanud, klient-server arhitektuuri toetavad programmeerimiskeeled.

Valikust jäetakse välja keeled, millega autoril varasem kogemus puudub, sest uue keele süntaksi, mustrite ja eelkõige korrektse rakendamise õppimisele kuluv aeg on diplomitöö kontekstis äärmiselt ebapraktiline.

Microsofti arendatav C# on kaasaegne objektorienteeritud programmeerimiskeel, mis võimaldab luua .NET raamistiku kuuluvaid rakendusi [13]. Süntaktiliselt on C# sarnane teistele laialt levinud objektorienteeritud programmeerimiskeeltele, nagu näiteks Java ja C++ [13].

C# üheks suureks eeliseks on võimekas päringukeel LINQ [14]. Lisaks iseloomustab keelt paralleelprogrammeerimise tugi asünkroonsete operatsioonide näol [13].

Java on Sun Microsystems poolt loodud objektorienteeritud dünaamiline programmeerimiskeel, mis on ühtlasi üks populaarsemaid [15] programmeerimiskeeli maailmas [16]. Kuigi tänapäeval kastutatakse Javat eelkõige veebirakenduste jaoks, on keel kasutatav erinevateks eesmärkideks [17].

Java eeliseks on selle platvormist sõltumatus – Java rakendused töötavad identselt sõltumata süsteemist, millel neid käivitatakse [17].

Python on populaarne [15] objektorienteeritud programmeerimiskeel, mille eelisteks on lihtne süntaks, modulaarsus ning äärmiselt kiire arenduste esmane testimine kompileerimise puudumise tõttu [18]. Pythoni veebirakenduste arenduses on välja kujunenud kaks põhilist raamistikku: Flask ja Django.

PHP on veebiarenduses levinud avatud lähtekoodiga skriptimiskeel [19]. PHP rakendus viib kõik protsessid läbi serveris ning saadab kliendile vastuseks vaid vajaliku HTML faili, mille abil saab peita rakenduse koodi kasutaja eest [19].

Valikus olevate programmeerimiskeelte omavahelisel võrdlemisel (Tabel 1) on arvesse võetud autori hinnangut autori keele rakendamise kogemuse ning õppimise keerukuse kohta. Ühe komponendina on välja toodud programmeerimiskeelte vastavus eelnevalt sõnastatud eeldustele.

Tabel 1. Programmeerimiskeelte võrdlustabel

Programmeerimiskeel	**Kogemus**	**Õppimise keerukus**	**Nõuetele vastavus**
C#	Väga hea	Keskmine	Väga hea
Java	Hea	Keskmine	Hea
Python	Rahuldav	Madal	Hea
PHP	Rahuldav	Madal	Keskmine

C# vastab kõikidele tehnoloogia valimiseks sätestatud lähtepunktidele: autoril on keelega eelnev põhjalik kogemus, keel sisaldab ohtralt baasfunktsionaalsust ning funktsionaalsust lisavaid mooduleid on lihtne kasutusele võtta NuGet [20] paketihalduri abil. Lisaks on Registrite ja Infosüsteemide Keskusel .NET raamistikul töötavatele rakendustele X-teel suhtlemise toetamiseks loodud standardmoodul [21].

Lähtuvalt C# programmeerimiskeele valikust põhineb arendatav rakendus .NET platvormil. .NET platvorm on avatud lähtekoodiga arendusplatvorm, mis annab arendajatele vahendid erinevatele seadmetele rakenduste loomiseks.

.NET platvorm kujutab endast Windowsi rakenduste arendamisele suunatud .NET Framework'i, platvormiülesele arendusele suunatud .NET Core ning iOS ja Android rakenduste arendamiseks mõeldud Xamarini [22]. Kuna Xamarin on mõeldud mobiilirakenduste arendamiseks on diplomitöö kontekstis aktuaalsed vaid .NET Core ning .NET Framework.

2020. aasta novembris andis .NET platvormi arendaja Microsoft välja NET 5.0 [23], mille eesmärk on ühendada kõik .NET platvormi pakutavad võimalused ühe raamistiku alla [24]. .NET Core'le sarnaselt ei sõltu .NET 5.0 rakenduse käivitamisel kindlast platvormist [23].

Lähtudes platvormist sõltumatuse eelisest tuleks valik teha .NET Core ja .NET 5.0 vahel. Diplomitöö kirjutamise ajal on .NET 5.0 võrdlemisi uus raamistik. Uute raamistike puhul võib probleeme esineda näiteks paketihalduris vajalike moodulite leidmisel, kuna uut raamistiku veel ei toetata. .NET Core 3.1 tuge on pikendatud 2022. aasta lõpuni [25] ning vajadusel on võimalik sooritada üleminek uuemale raamistikule. Lähtudes välja toodud eelistest ja piirangutest valitakse raamistikuks .NET Core 3.1.

4.4.2 Kasutajaliidese tehnoloogia

Kasutajaliidese loomisel on .NET Core raamistikus kaks erinevat võimalikku lähenemisviisi: serverist staatilisi vaateid tagastav rakendus ASP.NET MVC [26] põhimõtetel või serveripoolse API-liideste [27] ja eraldiseisva klientrakenduse loomine.

Alternatiividena eksisteerivad Razor Pages [28] ning selle edasiarendus Blazor [29], kuid ka MVC puhul on kasutusel mõlema tehnoloogia põhiliseks eeliseks olev Razor süntaks [30], mis võimaldab mugavalt rakendada C# funktsionaalsust HTML failides. Kompileerides lähtekoodi WebAssembly formaati, võimaldab Blazor C# rakenduse koodi käivitada brauseris [29]. Selline lähenemine on eelistatud veebirakendustes, kus on prioriteediks maksimaalse jõudluse saavutamine, nagu näiteks 3D mängud või visuaalne töötlus [31].

MVC mustril põhinev rakendus tagastab päringu korral serverist äriloogiliste protsesside alusel moodustatud vaate [26]. Mustri eesmärgiks on veebirakenduste erinevate loogiliste komponentide eraldamine ja nende iseseisva testimise võimalikkuse tagamine [26].

API-liideste puhul tagastatakse serverist vaid päringule vastavad andmed. Taolise metoodika puhul on eelduseks eraldiseisva klientrakenduse loomine, mille ülesandeks on vastavalt kasutaja tegevustele päringute sooritamine ja vastuste kuvamine kasutajale arusaadavas formaadis. Klientrakenduste loomisel on eelistatud vahenditeks erinevad JavaScripti raamistikud.

JavaScript on veebiarenduses laialt levinud programmeerimiskeel, mis võimaldab luua dünaamilise või interaktiivse sisuga veebilehti [32]. Keel annab arendajale võimaluse kasutajale kuvatavas vaates muudatusi teha läbi *Document Object Model'*i *ehk* DOM-i, mis on HTML ja CSS sisu manipuleerimiseks loodud liides [32].

Enne JavaScripti loomist kuvasid veebilehed põhiliselt staatilisi dokumente [33]. JavaScripti saabumisel hakkasid veebilehed üha enam võtma interaktiivset kuju, mida täna nimetatakse veebirakenduseks [33]. Suurema edasiminekuna tuleb välja tuua ka üheleherakenduste arengu, mille puhul kasutajale nähtavalt ei laeta uut lehte, vaid taustal teostavate päringute andmetele tuginedes uuendatakse kuvatavat vaadet [33].

Puhast JavaScripti kasutades nõuab interaktiivse ja muutuva sisuga veebirakenduse loomine JavaScripti väga põhjaliku tundmist ning ohtralt programmeerimist. Oluline on

silmas pidada ka veebibrauserite mitmesuguseid erisusi JavaScripti alusfunktsioonide toetamisel.

Nende probleemide lahendamiseks on viimase kümnendi jooksul välja töötatud mitmesugused JavaScripti raamistikud, nagu näiteks Vue.js, Angular.js ja React.js [33]. Raamistikud lihtsustavad oluliselt veebiarendust ning seavad rakenduse loomisele kindlad raamid, mis panustavad rakenduse töökindlusesse selle kasvades [33].

Lähtudes diplomitöö skoobist on eraldiseisva klientrakenduse loomine ebapraktiline. Klientrakendus nõuab toimimiseks kindlate alusfunktsionaalsuste loomist ja eelnevalt seadistamist, mis kujutab endast täiendavat tööd ja ajakulu. Korrektse arhitektuuri loomisel ja säilimisel ei nõua MVC mustrilt üleminek klientrakenduse loomist võimaldavatele API-liidestele palju täiendavaid muudatusi. Tuginedes kõigele eelnevale on diplomitöö kasutajaliidese tehnoloogiaks ASP.NET MVC.

4.4.3 Andmebaasi analüüs

Prototüüprakendus peab peale valmimist võimaldama algset kasutuselevõttu menetlusprotsessi toetava tööriistana. Analüütikutele peab võimaldama töö jätkamist ka prototüüplahendusele eelnenud meetoditega. Sellise lähenemise võimaldamiseks võetakse rakenduse töötamiseks vajalikud algandmed sisendiks soojusvaldkonna taotluse ja võrdlusanalüüsi Microsoft Exceli failidest. Loodav lahendus võimaldab edasisel arendusel vajadusel üleminekut relatsioonilisele andmebaasile nagu eelnevalt rakenduse sisemise arhitektuuri (Lisa 3) analüüsis välja toodi.

Tabelitest andmete pärimiseks ja salvestamiseks on arendusele eelnevalt oluline valida Microsoft Excel failidega liidestumist võimaldav tarkvaramoodul. Valim moodustatakse .NET platvormi paketihalduris NuGet kättesaadavatest ja kirjeldatud funktsionaalsust pakkuvatest teekidest.

Diplomitöö puhul on oluline, et teek või erinevate teekide kombinatsioon pakuks autorile võimaluse Excel faili avada, sellest andmeid lugeda ning faili andmeid kirjutada ja salvestada. Kasutatav teek peab olema vabatarkvara.

EPPlus on Exceli töötlemisega seotud teekidest NuGet paketihalduris üks alla laetumaid [34]. Teek võimaldab kõiki prototüübi loomiseks vajalikke protsesse. Kõnealuse teegi

puhul on tegemist tasulise [35] tarkvaraga, mistõttu on diplomitöös selle teegi kasutamine välistatud.

NPOI on tarkvaraarendaja Tony Au .NET platvormile porditud avaliku lähtekoodiga versioon Java NPOI teegist [36]. Teek võimaldab .xlsx, .xls, .docx failiformaatide lugemist ja neisse kirjutamist. NPOI teegist on omakorda loodud .NET Core versioon, kuid selle arendamine ja tugi on peatatud autoriõigustest tuleneva konflikti tõttu [37].

Sarnase funktsionaalsusega on ka teek ClosedXML, mis pakub Exceli jaoks suunatud lihtsustatud liidest [38] Microsoft Office dokumentide redigeerimiseks mõeldud [39] teegi Open XML SDK rakendamiseks.

LinqToExcel on tabelites LINQ süntaksi kasutades päringute sooritamist võimaldav [40] teek, mille autoriks on tarkvaraarendaja Paul Yoder. Samuti võimaldab teek märgendite abil objekti erinevate väljade sidumine kindla tabeli väljaga [40]. Teegi puhul on tegemist MIT litsentsi alusel pakutava avaliku lähtekoodiga vabatarkvaraga [40].

Valik taandub NPOI .NET Core variandi ja ClosedXML vahele, sest EFPlus on tasuline tarkvara ning vaatamata teegi pakutavatele mugavustele ei ole LinqToExcel puhul võimalik dokumenti andmeid kirjutada. ClosedXML pakub aga MVC ja WebAPI veebirakenduste jaoks laiendeid [38], millega on faili loomine ja alla laadimine lihtsustatud. Ajutises testkeskkonnas mainitud teekide liideseid võrreldes tundus autori jaoks ClosedXML liides mugavam ja selgesõnalisem. Kõigele eelnevale tuginedes valitakse käesolevas diplomitöös Excel failiformaadiga töötamiseks ClosedXML teek.

4.4.4 Versioonihaldustarkvara valik

Versioonihaldussüsteemi eesmärk on tarkvara lähtekoodi jagamine arendusprotsessis osalevate arendajate vahel, võimaldades seeläbi koostööd [41], [42]. Lisaks talletab lähtekoodi uuendamisel versioonihaldustarkvara uuendusele eelnenud versiooni, andes ülevaate lähtekoodi eelnevatest olekutest ja ajaloost.

Versioonihaldussüsteemid jagunevad arhitektuuri alusel kaheks: keskse (*Centralized Version Control Systems)* ja hajusa (*Distributed Version Control Systems)* arhitektuuriga versioonihaldussüsteemideks [41].

Keskse arhitektuuriga versioonihaldussüsteemid olid aastaid tarkvaraarenduses standardiks [41]. Sellise lähenemisega tuntuimateks tarkvaradeks on Subversion ja CVS [41], [43], [44]. Keskse arhitektuuriga versioonihaldussüsteemi puhul on probleemiks lähtekoodi asumine ühes keskses serveris, sest serveri- või kettatõrgete puhul on lähtekoodi käideldavus tugevalt mõjutatud [41].

Taoline nõrkus hajusa arhitektuuriga versioonihaldustarkvaradel puudub, sest lokaalselt hoitakse koopiat kesksest repositooriumist [41]. Tuntuimad keskse arhitektuuriga versioonihaldussüsteemid on Git ja Mercurial. Versioonihaldussüsteemi valikul lähtub autor eelnevast kogemusest (Tabel 1), versioonihaldussüsteemi arhitektuurilistest eelistest ja tööandja arendusprojektides kasutusel olevatest võimalustest.

Tabel 2. Autori kogemus valikus olevate versioonihaldustarkvaradega

Tarkvara	**Autori kogemus**
Git	Hea
Mercurial	Puudub
Subversion	Väga hea

Mercurial ei ole Registrite ja Infosüsteemide Keskuses töövahendina ei kasutusel, seetõttu taandub valik SVN'i ja Git'i vahele. Arvestades toodud eeliseid ja piiranguid ning autori soovi teadmisi süvendada, kasutatakse käesolevas diplomitöös versioonihaldussüsteemina Git'i. Lisaks on versioonihaldusel oluline silmas pidada järgnevaid häid tavasid:

- *Commit* peaks sisaldama ainult omavahel seotud muudatusi [45].
- *Commit* ei tohiks olla sisult väga mahukas, et seda oleks hiljem lihtne lugeda ja muudatusi mõista [45], [46].
- *Commit'*i nimi või kirjeldus peaks võimalikult täpselt välja näitama tehtud muudatusi [45], [46].
- *Commit* ei tohiks halvata rakenduse tööd või selle käivitamist [45].

4.5 Testimisplaan

Testimise aspektist on diplomitöö põhifookuses rakenduse äriloogika. Äriloogika testimiseks kaetakse seda hõlmav funktsionaalsus automaattestidega, mis võimaldavad hinnata rakenduse erinevate osade vastavust nendele seatud eeldustele. Seeläbi annavad automaattestid arendajale märku juhtudest, mil rakenduse lähtekoodi muudatuse tulemusena ei vasta enam ootustele kindla mooduli tööprotsess või selle väljund.

Diplomitöös loodava prototüübi puhul on automaattestide kirjutamine vajalik, sest testimisega tagatakse menetluses rakendatavate protsesside täpsus ja töökindlus. Lisaks annavad testid tulevikus rakenduse lähtekoodi muutvatele arendajatele julgust muudatusi teha ning võimaldavad vajadusel rakenduse lähtekoodi struktuuri põhjalikumat muutmist ehk refaktoreerimist [47].

Testimistel kasutatavateks töövahenditeks valib autor oma senisele töökogemuse tuginedes NUnit [48] testimise raamistiku koos objektide funktsionaalsuse imiteerimist võimaldava raamistikuga Moq.

4.6 Analüüsi kokkuvõte

Diplomitöö eesmärk on luua prototüüptarkvara, mis võtab sisendiks Microsoft Excel formaadis soojusvaldkonna hinnataotluse ning annab andmeanalüüsi ja arvutuste alusel väljundiks andmed, mis abistavad Konkurentsiameti analüütikut hinnataotluse kooskõlastamisel. Tarkvara nõuded sõnastati lähtuvalt kasutaja vajadusest ja visioonist.

Rakenduse analüüsi käigus valiti tehnoloogiad, millega alustatakse lähteülesande lahendamist. Tehnoloogiate ja arendustehnikate valimisel lähtuti eelkõige vahendi sobivusest ülesande lahendamiseks, samas võttes arvesse valikupõhiseid eeliseid ja sellega kaasnevaid piiranguid.

Prototüüplahendusena luuakse arhitektuuriliselt kolmest kihist koosnev MVC mustril põhinev veebirakendus, mille arendamisel kasutatakse C# programmeerimiskeelt ning sellega seotud .NET platvormi raamistiku .NET Core 3.1. Programmeerimiskeel C# osutus valituks selle eeliste ja autori tugeva kogemuse tõttu.

Rakenduse andmeallikaks saavad olema Konkurentsiameti igapäevases tööprotsessis kasutavad võrdlusanalüüsi Microsoft Exceli fail ja samas failiformaadis soojusvaldkonna

taotlus. Rakenduse arhitektuuri kavandamisel võeti arvesse tulevikus relatsioonilise andmebaasi kasutuselevõtu võimaldamist. Andmeallikatega liidestumiseks kasutatakse diplomitöös teeki ClosedXML.

Arenduse vältel kaetakse rakenduse äriloogika automaattestidega, et veenduda äriprotsessi täpsuses ja töökindluses.

Prototüübi arenduseks kasutatavaid arendustööriistu analüüsis ei kajastatud, sest arendusvahendite valik sõltub enamasti ettevõttes kasutusel olevatest vahenditest. Arendusest võimalikult täpse ülevaate saamiseks tuuakse välja prototüüprakenduse arendamiseks kasutatavad töövahendid.

Prototüübi arendamiseks kasutatakse integreeritud programmeerimiskeskkonnana Microsoft Visual Studio Professional 2019 tarkvara koos otsingut, veatuvastust ning muid mugavusfunktsioone pakkuva liidesega JetBrains ReSharper. Arendamisel kasutatakse versioonihaldustarkvarana Git'i.

5 Arendus

Käesolevas peatükis käsitletakse prototüübi arendusprotsessis esinenud põhilisi väljakutseid ja rakendatud metoodikaid, et anda põhjalik ülevaade rakenduse loomisest ja selle käigus langetatud otsustest.

5.1 Äriloogika

Prototüübi äriloogika jaguneb põhiliselt kolmeks eraldiseisvaks osaks:

1. Tabelitest taotluste andmete rakendusse lugemine.
2. Soojuse piirhinna ja sellega seotud erisuguste näitajate arvutused.
3. Arvutuste tulemuste fikseeritud statistilistele näitajatele vastavuse kontroll.

5.1.1 Andmete lugemine

Taotlusest andmete lugemisel kujunes põhiliseks väljakutseks andmete asukoha määramine juhtudel, kus andmete asukoht pole kindla rea ja veeruga määratud. Probleemi lahendamist lihtsustas märkimisväärselt taotluse enamjaolt punktuaalne struktuur, mille puhul on kindlad andmeväljad seotud kindla järjekorranumbriga fikseeritud veerus. (Joonis 3).

2	**Kasutatud kütuste bilanss:**			
2.1	**Kütus 1:**			
2.1.1	Kütuse liik	-		
2.1.2	Kasutatud kütuse kogus	*(ühik)*	0,0	0,0
2.1.3	Kütuse keskmine küttevääртus	*MWh/(ühik)*	0,00	0,00
2.1.4	Kütuse energiasisaldus (primaarenergia)	MWh	0	0
2.1.5	Tootmismaht kütusega (võrku antud soojus)	MWh	0	0
2.1.6	Keskmine tootmise kasutegur antud kütusega	%	0,00	0,00
2.2	**Kütus 2:**			
2.2.1	Kütuse liik	-		
2.2.2	Kasutatud kütuse kogus	*(ühik)*	0,0	0,0
2.2.3	Kütuse keskmine küttevääртus	*MWh/(ühik)*	0,00	0,00
2.2.4	Kütuse energiasisaldus (primaarenergia)	MWh	0	0
2.2.5	Tootmismaht kütusega (võrku antud soojus)	MWh	0	0
2.2.6	Keskmine tootmise kasutegur antud kütusega	%	0,00	0,00

Joonis 3. Näide taotluse struktuurist

Taolist lähenemist on kasutatud peamiselt töölehtedel, milles olemite arv võib olla varieeruv. Järjekorranumbri alusel on võimalik eristada olemit, selle indeksit ning kõnealuse olemi kindlat andmevälja.

Kirjeldatud järjekorranumbri struktuurist kujunes välja üldine algoritm järjekorranumbriga andmete lugemiseks, mis jaguneb järgmisteks sammudeks:

1. Leia kõik töölehelt otsitava olemit tuvastavale numbrile vastavad read
2. Määra leitud ridade arvu alusel instantside arv
3. Filtreeri ainult otsitavale instantsile vastavad read
4. Leia instantsi ridade hulgast otsitavale andmeväljale vastav rida

5.1.2 Andmete asukoha määramine

Fikseeritud asukohaga andmete leidmiseks loodi C# atribuudid, millega on võimalik andmeväljaga siduda andmete asukohaga seonduvat informatsiooni. Taolise lahenduse puhul välditakse andmete leidmiseks vajaliku info korduvat kirjutamist programmi erinevatesse punktidesse.

Pikemas perspektiivis võimaldab andmete taotlusest lugemise loogika üldistamine selle loogika eraldamist eraldiseisva teegina, mida oleks võimalik teiste valdkondade taotluste puhul rakendada.

```
[WorksheetMeta(Name="B. Algandmed")]
public class Algandmed : WorksheetEntity<SoojusHinnataotlus>
{
    [StaticWorksheetValue(Column = 3, Row = 2)]
    public IXLCell VorgupiirkonnaNimetus { get; set; }

    [StaticWorksheetValue(Column = 3, Row = 3)]
    public IXLCell EttevotteLiik { get; set; }

    [StaticWorksheetValue(Column = 5, Row = 3)]
    public IXLCell MajandusaastaAlgus { get; set; }

    [StaticWorksheetValue(Column = 5, Row = 4)]
    public IXLCell MajandusaastaLopp { get; set; }

    public List<Tootmisseade> Tootmisseadmed { get; set; }

    public List<Kutus> Kutused { get; set; }

    public List<MuugiAla> Muugialad { get; set; }

    public MuugiUldAndmed MuugiUldAndmed { get; set; }
}
```

Joonis 4. *WorksheetMeta* ja *StaticWorksheetValue* atribuutide rakendamine

Joonisel 4 rakendatakse eelmainitud atribuutide näiteid *WorksheetMeta* ja *StaticWorksheetValue*, mille abil lisatakse objektile seotud töölehe leidmiseks vajalikud andmed ning objekti väljadele seotud andmete asukoht. Töölehte tähistavat objekti ei oleks olnud mõistlik siduda kindla indeksiga töölehtede koguarvust, sest taotlust esitav ettevõtja võib eraldi töölehel esitada taotlust toetavaid andmeid. Samuti võib töölehe asukoht tahtmatult muutuda näiteks ümber tõstmise tagajärjel.

Töölehel korduvates andmestruktuurides (Joonis 3) asukohaga andmeväljade leidmiseks rakendati atribuute *WorksheetEntity* ja *WorksheetEntityValue*, mille abil lisati klassidele andmeväljade leidmiseks vajalikud metaandmed, nagu näiteks klassi või andmevälja tuvastav märgend või veeru number, millest märgendite abil loodud sõne alusel vastet otsima hakatakse.

Otsitavale andmeväljale vastav rida leitakse tabelist sõltuvalt andmevälja sisaldava andmestruktuuri esitamise viisist kindlal töölehel. Joonise 3 näitel kujutab järjekorranumber 2.1.3 endast soojuse tootmiseks kasutatud kütuse andmekogumit (2), selle esimest instantsi (1) ja kolmandat andmevälja „Kütuse keskmine küttevääртus“ (3).

```
[WorksheetEntity(IdentifierColumn = 1, Identifier = "2")]
public class KutuseAndmed : ExcelEntity<Algandmed>
{
    public IXLCell Majandusaasta { get; set; }

    [WorksheetEntityValue(Identifier = "2")]
    public IXLCell Kogus { get; set; }

    [WorksheetEntityValue(Identifier = "3")]
    public IXLCell KeskmineKuttevaartus { get; set; }

    [WorksheetEntityValue(Identifier = "4")]
    public IXLCell Energiasisaldus { get; set; }

    [WorksheetEntityValue(Identifier = "5")]
    public IXLCell Tootmismaht { get; set; }

    [WorksheetEntityValue(Identifier = "6")]
    public IXLCell Kasutegur { get; set; }
}
```

Joonis 5. *WorksheetEntity* ja *WorksheetEntityValue* atribuutide rakendamine

Illustreerimaks järjekorranumbri loomiseks vajalike andmete lisamist andmesiirdeobjektile on Joonisel 5 näitena välja toodud tootmisprotsessis kasutatava kütuse andmeid (Joonis 3) väljendav klass.

Domeeniobjektide eraldatuse ja sõltumatuse tagamiseks loodi taotluse andmete programmi lugemiseks eraldiseisvad andmesiirdeobjektid, mis sisaldavad eelnevates paragrahvides kirjeldatud erinevaid andmete asukoha määramise atribuute.

5.1.3 Atribuutide informatsiooni kasutamine

Andmestruktuurides andmete asukoha tähistamiseks rakendatud atribuutide andmetele ligi pääsemiseks (Joonis 6) kasutati C# programmeerimiskeele *reflection* metoodikat. *Reflection* annab dünaamilise ligipääsu andmetüüpide erinevatele osadele, nagu näiteks väljad või meetodid [49].

Ligipääs atribuutide andmeid tagastavatele staatilistele meetoditele on üksnes *extension method'*itel, mille abil töölehelt, tabeli reast või tabeli andmeid otsitakse.

```
private static WorksheetEntityValue GetPropertyAttribute<T>
(string propertyName)
{
    PropertyInfo propertyInfo = GetPropertyInfo<T>(propertyName);

    var attribute = propertyInfo.
          GetCustomAttribute<WorksheetEntityValue>();

    if (attribute is null)
    {
          throw new AttributeNotFoundExeption
                (typeof(WorksheetEntityValue), typeof(T));
    }

    return attribute;
 }
```

Joonis 6. Atribuutide väärtuste kasutamine *reflection*'i abil

Extension method'id, mille abil andmeväljale otsitav väärtus leitakse, on loodud erinevatele ClosedXML teegi klassidele erinevate olukordade jaoks, nagu näiteks eelnevalt leitud instantsiga seotud ridade hulgast andmevälja väärtuse otsimine (Joonis 7). Kirjeldatud *extension method*'ite kasutusjuhud ja sisu on rakenduse lähtekoodis dokumenteeritud, mistõttu on autori arvates rakenduse võimalikel tulevastel arendajatele andmete lugemiseks kasutatav metoodika kergesti hoomatav. Dokumenteerimiseks kasutati C# programmeerimiskeele [50] XML dokumentatsiooni formaati.

```
public static IXLCell FindMappedCell<T> (this List<IXLRow> rows,
string propertyName) where T : BaseExcelEntity
{
    var mapping = GetPropertyAttribute<T>(propertyName);

    var entityMapping = GetEntityMapping<T>();

    var matchingRow = rows.FirstOrDefault(x =>
         x.Cell(entityMapping.IdentifierColumn)
        .GetString().Contains(mapping.Identifier));

    return matchingRow?.Cell(mapping.Column);
 }
```

Joonis 7. Ridade hulgast andmevälja otsimine

5.1.4 Andmete valideerimine

Sisendiks oleva taotluse andmete valideerimine jaguneb kolme etappi.

Esmalt määratakse andmete asukoht kasutades andmesiirdeobjektile ja selle väljadele lisatud atribuute. Andmete leidmiseks kasutatakse eelnevalt kirjeldatud metoodikaid. Prototüübi edasisel arendusel oleks andmesiirdeobjektide atribuutidesse praktiline lisada ka tabelis oleva välja nimetus, et täielikult garanteerida tabelist leitud rea vastavust otsitavale andmeväljale. Kui otsitavat rida ei leita, ei tekitata rakenduses veaolukorda vaid tagastatakse tühi väärtus.

Valideerimise teine samm kujutab endast andmesiirdeobjektide andmete ümbertõstmist domeeniobjektidesse, mille puhul kontrollitakse, kas andmed on vastavuses domeeniobjekti välja andmetüübiga. Selles etapis eraldatakse andmetest näiteks tühjad väljad või sootuks täiesti tühjaks jäänud objektid.

Kolmandana valideeritakse domeeniobjektide andmetest saadud tulemusi vastavalt ärireeglitele. Selleks moodustatakse eelnevalt erinevate domeeniobjektide väärtuste põhjal tehtud arvutuste tulemusi väljendavad objektid. Need objektid sisaldavad näiteks tootmise või kütuste üldandmeid erinevatel majandusaastatel.

```
    public class ElekterFiscalStatisticsValidator :
    AbstractValidator<ElekterFiscalStatistics>
    {
        public ElekterFiscalStatisticsValidator(ElekterStatistics parent)
        {
            RuleFor(x => x.Erikulu).Must(x => x <= parent.Allowed.Erikulu)
                .WithErrorCode("A")
                .WithMessage(model => $"Elektrienergia erikulu erineb lubatud
määrast. (Taotlus: {Math.Round(model.Erikulu, 2)} Lubatud:
{Math.Round(parent.Allowed.Erikulu, 2)})")
                .Must(x => x <= parent.Maximum.Erikulu)
                .WithErrorCode("M");

            RuleFor(x => x.Kogus)
                .Must(x => x <= parent.Allowed.Kogus)
                .WithErrorCode("A")
                .WithMessage(model => $"Elektrienergia kogus erineb lubatud
määrast. (Taotlus: {Math.Round(model.Kogus, 2)} Lubatud:
{Math.Round(parent.Allowed.Kogus, 2)})")
                .Must(x => x <= parent.Maximum.Kogus)
                .WithErrorCode("M");
        }
    }
```

Joonis 8. FluentValidation teegi kasutamine tulemusobjekti valideerimiseks

Etapis kontrollitakse nende andmete vastavust erinevatele näitajatele või andmetele seatud standarditele, mis on äriloogiliselt prototüübi funktsionaalsuse üheks põhiliseks osaks. Andmete valideerimiseks kasutatakse rakenduses .NET teeki FluentValidation, mis võimaldab [51] andmeväljadele lihtsasti arusaadava süntaksiga (Joonis 8) valideerimisreegleid lisada.

5.2 Andmeallikad

Prototüübi üheks põhiliseks mittefunktsionaalseks nõudeks on võimaldada prototüübi kasutamist nii, et see võimaldaks senise käsitsi läbi viidava tööprotsessi jätkamist. Selle eesmärgi saavutamiseks ei saa prototüüprakendus olla sisendandmetega pidevas seoses digitaalse andmebaasi näol, vaid andmeallikad tuleb prototüübile sisendiks anda.

Prototüübile sisendiks antud võrdlusanalüüsi ja taotluse sisendfailid salvestatakse serverisse fikseeritud formaadis nimega, mis sisaldab sessioonivõtit. Sessiooni jooksul tehtud päringute täitmiseks võetakse esmalt kaustast sessioonile vastavad failid ning luuakse *object relational mapping* tarkvarale sarnaselt domeeniobjektide kogum, millest hiljem spetsiifiliste päringute abil andmeid leitakse.

Võrdlusanalüüsi faili andmetele ligipääsemiseks rakendatakse *repository* arendusmustrit, mis võimaldab tulevikus kerget üleminekut relatsioonilisele andmebaasile.

Taotluse andmete lugemisel ei rakendata sarnast metoodikat, sest selleks oleks eelnevalt vaja läbi viia taotluse andmete äri- ja süsteemianalüüs. Analüüsi käigus tuleks välja selgitada, millises formaadis on taotluse andmeid kõige praktilisem hoiustada – andmebaasis või prototüüplahendusele sarnaselt failidena. Prototüüpimise faasis taoliste detailide välja selgitamine ning vajaliku infrastruktuuri juurutamine oleks autori arvates liialt ennatlik ja ebapraktiline.

Kaust, milles parajasti serveris faile ajutiselt hoiustatakse, on määratud rakenduse konfiguratsioonis. Kausta sisu liigselt suureks kasvamise vältimiseks on rakendusse loodud vastav kontroll, mis eemaldab uute sisendfailide lisamise käigus kaustast kõik failid, mille loomise kuupäev ei ole sisendfailidega võrdne.

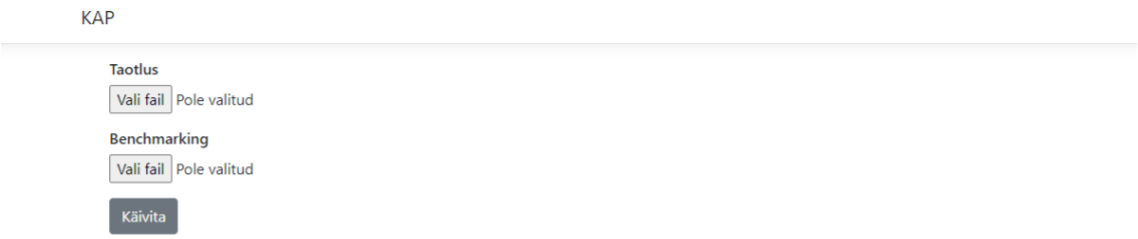
Prototüübi edasisel täiustamisel annab taoline lähenemine võimaluse hajutada rakenduse väljundit eraldi lehtedele, et edendada kasutajamugavust ja loetavust ja vajadusel funktsionaalsust laiendada.

5.3 Vaated

Kasutajaliidese puhul ei pandud rõhku eelnevale analüüsile, sest tegemist on prototüübiga. Lisaks ei olnud esialgu selget ja ühtset arusaama väljundist ja selle sisust, mistõttu väljundi lõplik formaat kujunes välja arenduse käigus Konkurentsiametiga toimunud arutelude jooksul.

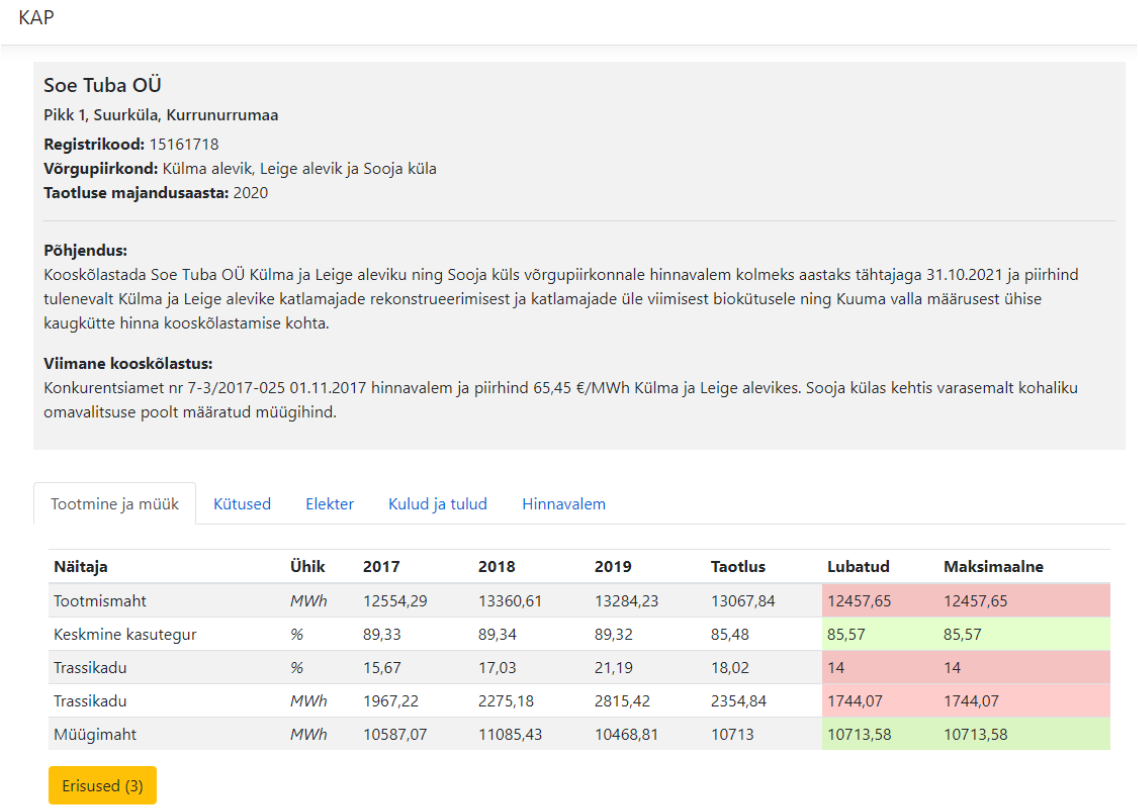
Prototüüprakenduses on kaks põhilist vaadet: sisendandmete vaade ning väljundi vaade.

Sisendandmete vaates (Joonis 9) saab kasutaja rakendusele sisendiks anda kõik rakenduse tööks vajalikud andmeallikad. Peale vajalike failide lisamist on kasutajal võimalik andmete töötlemise protsess käivitada.



Joonis 9. Kuvatõmmis prototüübi sisendvaatest

Väljundi vaates (Joonis 10) saab kasutaja ülevaate taotluse andmetest ja nende töötlemisel leitud ebakõladest**.** Kasutajale kuvatakse taotluse andmete põhjal loodud andmekogumid, mis on kategoriseeritud erinevatesse tabelitesse. Vaates kuvatakse näiteks ettevõtja tootmisandmeid, tootmises kasutatud kütuste andmeid ja ettevõtte majandustegevusega seotud näitajaid. Tabelites kuvatakse andmeid välja majandusaastate kaupa ning iga veeru lõpus on välja toodud vastava näitaja lubatud ja maksimaalsed piirid.



Joonis 10. Kuvatõmmis prototüübi väljundvaatest

Maksimaalset ja lubatud piiri kuvavad lahtrid on värvitud kas punaseks või roheliseks sõltuvalt sellest, kas kirjeldatav näitaja taotletaval majandusaastal vastab sisendandmete põhjal leitud väärtusele. Näitajad, mille puhul rakendus vajalike arvutusi sooritada ei suutnud või mille jaoks võrdlusandmeid ei leitud, kujutatakse kollasena.

Sellise lähenemisega saab kasutaja kiire ülevaate sellest, mis erisused taotluse andmete ja piirarvude vahel leiti. Iga kategooria tabeli all kuvatakse kasutajale valideerimisega seotud teateid, mis annavad kasutajale detailsemat informatsiooni vastava kategooria näitajate analüüsimisel leitud vigade või erisuste kohta.

Vaadete loetavuse ja taaskasutatavusest lähtudes jagati vaadetes kasutatavad tabelid eraldiseisvateks komponentideks. Vaadete alamkomponentideks jagamiseks kasutatakse ASP.NET Core MVC puhul *partial view*'sid [52].

5.4 Testimine

Automaattestide loomise vajadus võib esialgu taolise funktsionaalse prototüübi loomisel tunduda ebavajalik, kuid lähtudes seatud eesmärgist eduka prototüübi korral selle arendamist jätkata, tuleks rakenduse kriitilisemad osad võimalikus mahus automaattestidega katta.

Rakenduses on kokku 35 moodultesti, millest valdav enamus on suunatud taotluse andmete lugemise ja väljundiga seotud arvutuste testimiseks.

```
[Test]
public void FindsStaticValuesFromWorksheet()
{
    var firstWorksheet = GetFirstTestWorksheet();

    var subject = new FirstTestsheet();

    var staticCellOneExpected = 0.6969;
    var staticCellTwoExpected = 0.7878;

    subject.StaticCellOne = firstWorksheet.GetStaticCell
            <FirstTestsheet>(nameof(subject.StaticCellOne));
    subject.StaticCellTwo = firstWorksheet.GetStaticCell
            <FirstTestsheet>(nameof(subject.StaticCellTwo));

    Assert.IsNotNull(subject.StaticCellOne);
    Assert.IsNotNull(subject.StaticCellTwo);

    Assert.AreEqual(staticCellOneExpected,
    subject.StaticCellOne.GetDouble());
    Assert.AreEqual(staticCellTwoExpected,
    subject.StaticCellTwo.GetDouble());
}
```

Joonis 11. Moodultesti näide

Olulist rolli mängis rakenduse arendamise jooksul ka manuaalne testimine. Arendamise jooksul kontrolliti andmetöötluse tulemusi võrreldes neid taotluses väljatoodud andmete koondtulemustega. Esialgu rakendatakse prototüüpi senise tööprotsessiga paralleelselt, et veenduda reaalsete andmete põhjal taotluse töötlemisega seotud funktsionaalsuste korrektsuses ning leida andmete programmi lugemiseks loodud lahenduses kitsaskohti, millele tuleks arenduse jätkamisel rohkem tähelepanu pöörata.

5.5 Evitus

Prototüübile ligipääsu piiramiseks paigaldati rakendus Justiitsministeeriumi sisevõrku. Ligipääsu tagamiseks vaid määratud kasutajatele lisati rakendusele konfigureeritav võimekus kasutada autentimist Windows Active Directory põhjal, mis nõuab kasutajalt domeenikontoga sisse logimist enne rakenduse kasutamist.

6 Tulemused

Järgnevas osas annab autor hinnangu diplomitöö käigus valminud rakendusele. Hinnatakse rakenduse vastavust analüüsis seatud eesmärkidele ja nõuetele. Lisaks kirjeldatakse olulisi eesmärke lähitulevikus ning võimalike järgnevaid samme ja arhitektuurilisi lahendusi rakenduse edaspidisel arendamisel.

6.1 Valminud lahendus

Diplomitöö tulemusena valmis prototüüplahendus, mis võtab sisendiks soojusvaldkonna hinnataotluse ning pakub analüütikule esialgse väljundi, mille põhjal saab hinnataotluse osas otsuse langetada.

Rakendus loeb esmalt mällu taotluse andmed, millele järgneb andmetega seotud arvutuste sooritamine ja piirnäitajate leidmine. Peale rakenduse töö lõpetamist kuvatakse kasutajale väljund, mis koosneb taotluse andmestiku alusel arvutatud erinevatest kumulatiivsetest- ja võrdlusnäitajatest. Väljund on vastavalt andmete valdkonnale kategoriseeritud.

Töö käigus loodi Excel failidest kindla struktuuriga andmete lugemiseks baasfunktsionaalsused, mida saab hilisemalt kasutada rakenduse funktsionaalsuse laienemisel teistele valdkondadele või sootuks teistes projektides.

Prototüüplahenduse väljund ei lahenda täielikult püstitatud probleemi vaid demonstreerib probleemi lahendamiseks võimalikkust. Püstitatud probleemi täielikuks lahendamiseks on oluline prototüübi arendamise jätkamine, mille käigus optimeeritaks ja laiendataks rakenduse funktsionaalsust ning valmiks esmane tellija ootustele vastav rakendus.

Kui rakenduse arendamise järgnevate sammudega ei otsustata jätkata on rakendust siiski võimalik kasutada menetlusprotsessis abistava tööriistana, mille abil saab Konkurentsiameti kiire ülevaate taotluse andmestikust ja selle põhjal arvutatud näitajatest.

6.2 Nõuetele vastavus

Rakenduse arendamise käigus pandi rõhku eelkõige testitud ja töötava põhja loomisele, et tellija rahulolu korral projekti arendamise jätkamine oleks võimalikult efektiivne ning tööga jätkamine ei nõuaks rakenduse lähtekoodis liialt palju muudatusi. Seetõttu tehti tellijaga kokkuleppel nõuetes mõningaid järeleandmisi.

Näiteks loobuti esialgselt planeeritud võrdlusanalüüsi tabeli otsingu üleviimisest prototüüpi. Kuna rakendust peab saama senise tööprotsessiga paralleelselt kasutada, siis prototüübi arenduse käigus võrdlusanalüüsi failis olemasoleva funktsionaalsuse duplitseerimine ei olnud praktiline.

Samuti ei realiseeritud hinna kooskõlastamisel automaatset võrdlusanalüüsi faili täitmist taotlust kokkuvõtvate andmetega, sest eesmärk nõuab andmete sisestamist mitmesse erinevasse tabelisse, kus on Microsoft Exceli funktsioonid ja viited käsitsi peenhäälestatud. Sellest süveneb vajadus edasises arenduses teha üleminek relatsioonilisele andmebaasile võrdlusanalüüsi andmete hoiustamiseks.

Moodultestidega kaeti rakenduse kriitilisemad osad ning ülejäänud prototüübi funktsionaalsus kontrolliti manuaalse testimisega prototüübi arendamise käigus. Tuginedes testimise tulemustele võib väita, et kõik ülejäänud prototüübile sätestatud nõuded said edukalt täidetud.

Konkurentsiameti sõnul on väljatöötatud prototüübi abil võimalik hoida kokku analüütiku töö aega ning jagada olemasolevat tööjõuressurssi suurema majandusliku mõjuga ettevõtete andmete süvaanalüüsiks.

Kokkuvõttes saab väita, et prototüüp täidab selle loomiseks püstitatud esialgset eesmärki: prototüüp annab analüütikule väljundi, mille baasil on võimalik soojuse hinna kooskõlastamisel esialgseid otsuseid langetada.

6.3 Edasine arendus

Arendamise jätkamise suhtes langetatakse otsus peale diplomitöö esitamist.

Arendamise jätkamisel on ettenähtud:

- Arvutuste ja valideerimiste järkjärguline laiendamine vastavalt Konkurentsiameti soovidele ja tagasisidele.
- Üleminek relatsioonilisele andmebaasile hoiustamaks võrdlusanalüüsi ja kooskõlastamisega seotud andmeid.
- Statistikaametiga liidestumine vajalike võrdlusnäitajate pärimiseks.

6.3.1 Menetlusportaali loomine

Pikemas perspektiivis on lahenduse arendamise jätkamisel on ette nähtud järgmised sammud:

- Rakenduse võimekuse laiendamine teiste Konkurentsiameti hinnaregulatsiooni valdkondade toetamiseks.
- Konkurentsiametile veebipõhise menetluskeskkonna loomine, mis võimaldab ettevõtetel efektiivsemalt nii menetlusprotsessis osaleda kui ka ülevaadet saada.

Autori visiooni kohaselt arhitektuuriliselt on menetluskeskkonna loomiseks kaks võimalikku lähenemist: monoliitne arhitektuur [53] ja mikroteenuste arhitektuur [54], [55].

Mikroteenuste arhitektuuril (Lisa 5) on mitmesuguseid eeliseid monoliitse arhitektuuri (Lisa 4) ees:

- Rakendus ei sõltu funktsionaalsuselt ühest tehnoloogiast ja sellega kaasnevatest piirangutest.
- Iga valdkonna puhul võimalik valdkonnaspetsiifiline (*domain driven design*) lähenemine.

7 Kokkuvõte

Autori bakalaureusetöö eesmärgiks oli luua Konkurentsiametile prototüüp ettevõtete poolt esitatavate hinnataotluste menetlemise osaliseks automatiseerimiseks.

Tellija poolne esialgne visioon nägi lahenduseks ette kratti, mis tuginedes Konkurentsiameti poolt aja jooksul kogutud andmetele suudaks taotluseid ning nendega seotud andmeid analüüsida. Andmeanalüüsile ja seotud arvutustele põhinedes abistanuks planeeritav lahendus seni kõike eelnevat käsitööna teinud inimest ning vähendanuks taotlustest tulenevat töökoormust.

Prototüübi analüüsis sõnastati rakenduse eeldatav funktsionaalsus ning toodi välja ülesande lahendamiseks võimalikud töövahendid, mille hulgast selgelt põhjendatud valikud tehti. Samuti langetati analüüsi käigus rakenduse arhitektuuriga seotud otsused. Arendust käsitlevas osas tutvustati eelkõige lähteülesande lahendamiseks kasutatud metoodikaid koos põhjenduste ning illustreerivate näidetega.

Testimise tulemustele ja Konkurentsiameti tagasisidele tuginedes võib väita, et prototüüp suudab töö alguses sõnastatud lähteülesannet täita. Loodud prototüüp katab valdava enamuse analüüsis sõnastatud nõuetest ning kõik täitmata jäänud nõuded on diplomitöö lõpuosas põhjendatud.

Kasutatud kirjandus

[1] Riigikogu, „Kaugkütteseadus - KKütS,“ 2003. [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/103032017012. [Kasutatud 12 02 2021].

[2] Konkurentsiamet, „Konkurentsiamet - Soojusvaldkonna tutvustus,“ 2021. [Võrgumaterjal]. Available: https://www.konkurentsiamet.ee/et/vesi-soojus/soojus/valdkonna-tutvustus. [Kasutatud 12 02 2021].

[3] Konkurentsiamet, „Hindade kooskõlastamine,“ [Võrgumaterjal]. Available: https://www.konkurentsiamet.ee/et/vesi-soojus/soojus/hindade-kooskolastamine. [Kasutatud 12 02 2021].

[4] Konkurentsiamet, „Tarbijale ja/või võrguettevõtjale müüdav soojus,“ 2021. [Võrgumaterjal]. Available: https://www.konkurentsiamet.ee/sites/default/files/tarbijale_javoi_vorguettevotjale_muudav_soojus_2021.xlsx. [Kasutatud 12 02 2021].

[5] Konkurentsiamet, „Hinna- või järelvalvemenetluse läbiviimiseks vajalike andmete esitamise juhend soojusettevõtjale,“ 2017. [Võrgumaterjal]. Available: https://www.konkurentsiamet.ee/sites/default/files/1_hinna-_voi_jarelevalvemenetluse_labiviimiseks_vajalike_andmete_esitamise_juhend_soojusettevotjatele.pdf. [Kasutatud 12 02 2021].

[6] Konkurentsiamet, „Soojuse piirhinna kooskõlastamise põhimõtted,“ 2020. [Võrgumaterjal]. Available: https://www.konkurentsiamet.ee/sites/default/files/Soojus/soojuse_piirhinna_koosk_lastamise_p_him_tted.pdf. [Kasutatud 12 02 2021].

[7] University of Washington, „What is Excel?,“ University of Washington, 2021. [Võrgumaterjal]. Available: https://itconnect.uw.edu/learn/workshops/online-tutorials/microsoft-office-2010/microsoft-excel-2010/. [Kasutatud 12 02 2021].

[8] L. A. Pace ja K. A. Barchard, „Preventing human error: The impact of data entry methods on data accuracy and statistical results,“ *Computers in Human Behavior,* kd. 27, nr 5, pp. 1834-1839, 2011.

[9] Statistikaamet, „Palk ja tööjõukulu,“ [Võrgumaterjal]. Available: https://www.stat.ee/et/avasta-statistikat/valdkonnad/tooelu/palk-ja-toojoukulu. [Kasutatud 07 03 2021].

[10] Majandus- ja Kommunikatsiooniministeerium , „Energiatõhususe direktiiv,“ 16 10 2020. [Võrgumaterjal]. Available: https://www.mkm.ee/et/tegevused-eesmargid/energeetika/energiasaast. [Kasutatud 23 02 2021].

[11] M. Fowler, „Repository,“ [Võrgumaterjal]. Available: https://martinfowler.com/eaaCatalog/repository.html. [Kasutatud 22 02 2021].

[12] Riigi Infosüsteemide Amet, „Andmevahetuskiht X-Tee,“ 10 09 2020. [Võrgumaterjal]. Available: https://www.ria.ee/et/riigi-infosusteem/andmevahetuskiht-x-tee.html. [Kasutatud 04 03 2021].

[13] Microsoft, „.NET - A tour of the C# language,“ Microsoft, 28 Jaanuar 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 19 02 2021].

[14] Microsoft, „Language Integrated Query (LINQ),“ Microsoft, 02 02 2017. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/. [Kasutatud 19 02 2021].

[15] TIOBE, „TIOBE Index for February 2021,“ TIOBE, 02 2021. [Võrgumaterjal]. Available: https://www.tiobe.com/tiobe-index/. [Kasutatud 22 02 2021].

[16] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, D. Smith ja G. Bierman, „The Java Language Specification - Java SE 15 Edition,“ 10 08 2020. [Võrgumaterjal]. Available: https://docs.oracle.com/javase/specs/jls/se15/jls15.pdf. [Kasutatud 19 02 2021].

[17] H. Austerlitz, Data Acquisition Techniques Using PCs (Second Edition), Academic Press, 2003, pp. 326-360.

[18] Python Software Foundation, „What is Python? Executive summary,“ [Võrgumaterjal]. Available: https://www.python.org/doc/essays/blurb/. [Kasutatud 22 02 2021].

[19] The PHP Group, „What is PHP?,“ The PHP Group, 2001-2021. [Võrgumaterjal]. Available: https://www.php.net/manual/en/intro-whatis.php. [Kasutatud 22 02 2021].

[20] Microsoft, „What is NuGet?,“ Microsoft, 24 05 2019. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/nuget/what-is-nuget. [Kasutatud 26 02 2021].

[21] R. j. I. Keskus, „X-tee generaator,“ Registrite ja Infosüsteemide Keskus, [Võrgumaterjal]. Available: https://www.rik.ee/et/muud-teenused/x-tee-generaator. [Kasutatud 21 04 2021].

[22] Micrsosoft, „What is .NET?,“ [Võrgumaterjal]. Available: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet. [Kasutatud 07 03 2021].

[23] R. Lander, „Announcing .NET 5.0,“ Microsoft, 10 11 2020. [Võrgumaterjal]. Available: https://devblogs.microsoft.com/dotnet/announcing-net-5-0/. [Kasutatud 04 03 2021].

[24] R. Lander, „Introducing .NET 5.0,“ Microsoft, 06 05 2019. [Võrgumaterjal]. Available: https://devblogs.microsoft.com/dotnet/introducing-net-5/. [Kasutatud 07 03 2021].

[25] Microsoft, „Download .NET,“ [Võrgumaterjal]. Available: https://dotnet.microsoft.com/download/dotnet. [Kasutatud 04 03 2021].

[26] Microsoft, „Overview of ASP.NET Core MVC,“ Microsoft, 12 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/et-ee/aspnet/core/mvc/overview?view=aspnetcore-5.0. [Kasutatud 26 02 2021].

[27] Microsoft, „Create Web API-s with ASP.NET Core,“ Microsoft, 07 20 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-5.0. [Kasutatud 01 03 2021].

[28] Microsoft, „Introduction to Razor Pages in ASP.NET Core,“ 02 12 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-5.0&tabs=visual-studio. [Kasutatud 04 03 2021].

[29] Microsoft, „Blazor - Build client apps with C#,“ [Võrgumaterjal]. Available: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor. [Kasutatud 04 03 2021].

[30] Microsoft, „Razor syntax reference for ASP.NET Core,“ 02 12 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-5.0. [Kasutatud 04 03 2021].

[31] Mozilla, „WebAssembly Concepts,“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts. [Kasutatud 04 03 2021].

[32] Mozilla, „Mozilla Web Docs - What is JavaScript?,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Kasutatud 26 02 2021].

[33] Mozilla, „Introduction to client-side frameworks,“ Mozilla, [Võrgumaterjal]. Available: https://developer.cdn.mozilla.net/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction. [Kasutatud 26 02 2021].

[34] Microsoft, „NuGet Gallery - Search,“ [Võrgumaterjal]. Available: https://www.nuget.org/packages?packagetype=&sortby=totalDownloads-desc&q=Excel&prerel=false. [Kasutatud 04 03 2021].

[35] EFPlus Software AB, „EFPlus Software - Our license types,“ EFPlus Software AB, [Võrgumaterjal]. Available: https://www.epplussoftware.com/en/LicenseOverview/. [Kasutatud 11 03 2021].

[36] T. Qu, „GitHub - NPOI,“ [Võrgumaterjal]. Available: https://github.com/nissl-lab/npoi. [Kasutatud 11 03 2021].

[37] Y. Xiadong ja T. Qu, „GitHub - NPOI,“ [Võrgumaterjal]. Available: https://github.com/dotnetcore/NPOI. [Kasutatud 11 03 2021].

[38] M. de Leon, „GitHub - ClosedXML,“ [Võrgumaterjal]. Available: https://github.com/ClosedXML/ClosedXML. [Kasutatud 14 03 2021].

[39] Microsoft, „GitHub - Open XML SDK,“ [Võrgumaterjal]. Available: https://github.com/OfficeDev/Open-XML-SDK. [Kasutatud 14 03 2021].

[40] P. Yoder, „GitHub - LinqToExcel,“ [Võrgumaterjal]. Available: https://github.com/paulyoder/LinqToExcel. [Kasutatud 11 03 2021].

[41] S. Chacon ja B. Straub, Pro Git (2nd Edition), Apress, 2020.

[42] Atlassian, „What is version control?,“ Atlassian, 2021. [Võrgumaterjal]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control. [Kasutatud 18 02 2021].

[43] The Apache Software Foundation, „Apache Subversion,“ The Apache Software Foundation, 2018. [Võrgumaterjal]. Available: https://subversion.apache.org/. [Kasutatud 18 02 2021].

[44] D. R. Price, „Introduction to CVS,“ Ximbiot, Free Software Foundation, 2005-2006. [Võrgumaterjal]. Available: https://www.nongnu.org/cvs/. [Kasutatud 18 02 2021].

[45] Git Tower, „Version Control Best Practices,“ Git Tower, 07 2020. [Võrgumaterjal]. Available: https://www.git-tower.com/blog/version-control-best-practices/. [Kasutatud 22 02 2021].

[46] Atlassian, „Source code management,“ Atlassian, [Võrgumaterjal]. Available: https://www.atlassian.com/git/tutorials/source-code-management. [Kasutatud 22 02 2021].

[47] M. Fowler, „SelfTestingCode,“ 5 05 2005. [Võrgumaterjal]. Available: https://martinfowler.com/bliki/SelfTestingCode.html. [Kasutatud 04 03 2021].

[48] C. Poole ja R. Prouse, „What is NUnit?,“ 2019. [Võrgumaterjal]. Available: https://nunit.org/. [Kasutatud 04 03 2021].

[49] Microsoft, „Reflection (C#),“ 20 07 2015. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection. [Kasutatud 08 04 2021].

[50] Microsoft, „XML documentation comments,“ Microsoft, 20 07 2015. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/xmldoc/. [Kasutatud 21 04 2021].

[51] J. Skinner, „FluentValidation Docs - Creating your first validator,“ 08 08 2020. [Võrgumaterjal]. Available: https://docs.fluentvalidation.net/en/latest/start.html#. [Kasutatud 19 04 2021].

[52] Microsoft, „Partial views in ASP.NET Core,“ Microsoft, 07 06 2019. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial?view=aspnetcore-5.0. [Kasutatud 19 04 2021].

[53] C. Richardson, „microservices.io - Pattern: Monolithic Architecture,“ 2020. [Võrgumaterjal]. Available: https://microservices.io/patterns/monolithic.html. [Kasutatud 22 02 2021].

[54] M. Fowler, „Martin Fowler - Microservices,“ 14 03 2014. [Võrgumaterjal]. Available: https://martinfowler.com/articles/microservices.html. [Kasutatud 22 02 2021].

[55] C. Richardson, „Microservices,“ 2020. [Võrgumaterjal]. Available: https://microservices.io/. [Kasutatud 22 02 2021].

[56] Microsoft, „Attributes (C#),“ 26 04 2018. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/attributes/. [Kasutatud 22 03 2021].

Lisa 1 - Lihtlitsents

Mina, Jan-Erik Kalmus

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Konkurentsiameti soojusvaldkonna põhjendatud hinna kooskõlastamise tarkvara prototüüp" , mille juhendajad on Meelis Antoi ja Andreas Türk.

    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.

Lisa 2 – Soojuse piirhinna arvutamise valem

$$\boldsymbol{h} = \frac{\boldsymbol{kulud}}{\boldsymbol{Q}} + \frac{\boldsymbol{k_{kütus}}}{\boldsymbol{Q}} \times \boldsymbol{hind_{kütus}} \left[\frac{€}{\boldsymbol{MWh}}\right]$$

h – soojuse piirihind (€/MWh)

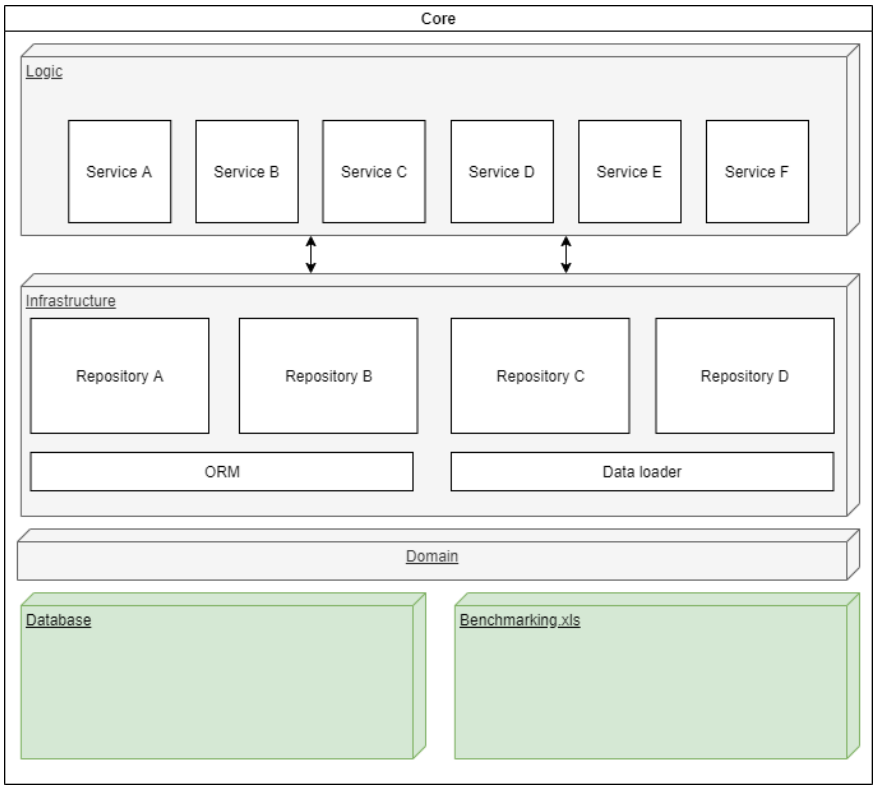
kulud – põhjendatud muutuv- ja tegevuskulud, kapitalikulu ning põhjendatud tulukus, v.a kulud kütusele ja/või sisse ostetud soojusele

$k_{kütus}$ - kütuse kogus

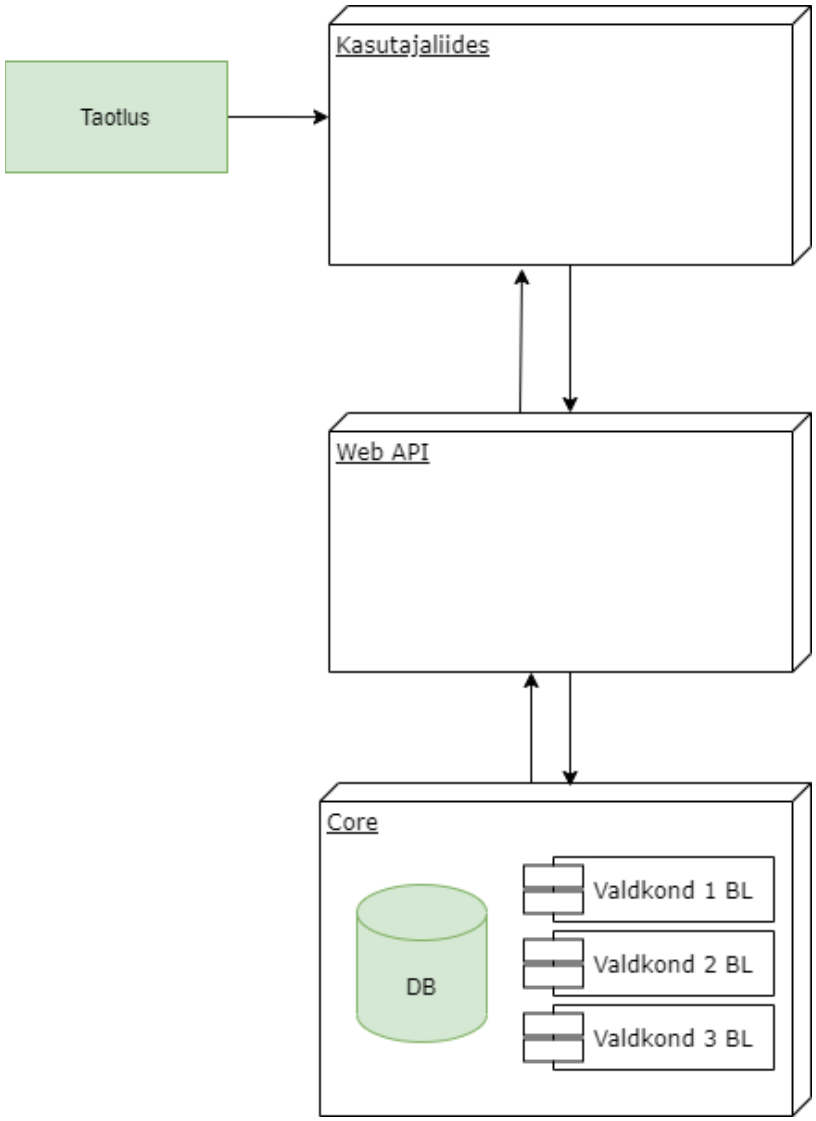
$hind_{kütus}$ – kütuse ostuhinna prognoos

Q – soojuse müügimaht

Lisa 3 – Prototüüptarkvara sisemine struktuur



Lisa 4 – Menetluskeskkonna arhitektuur monoliitrakendusena



Lisa 5 – Menetluskeskkonna arhitektuur mikroteenuste rakendamisel