TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ahto Karolin 192824IADB

Elektri börsihinna API prototüübi loomine

Bakalaureusetöö

Juhendaja: Meelis Antoi
MSc
Kaasjuhendaja Tiit Kuuskmäe
MSc

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: Ahto Karolin

2.01.2023

Annotatsioon

Antud bakalaureuse töö eesmärk on luua programmiliides tüüpi rakendus, mis realiseeriks erinevaid kasutuslugusid elektribörsi hinna tarbimiseks targa kodu lahendustes. Selliselt lahendab rakendus puudujäägi turul, kus Elering AS annab elektribörsi hinna andmed aga selleks, et neist saada kasu kulude kokkuhoiuks on igal kasutajal vaja teha andmetöötlust.

Käesolevas lõputöös on analüüsitud parimaid praktikaid ja erinevaid nõudeid, et saavutada toimiv rakenduse minimaalselt toimiv ja väärtust toov versioon. Kirjeldatud on realisatsiooni protsess kattes arenduse, testide ja evitus pilveteenuse majutuseks. Tulemusena valminud rakendus on Eestis avalikult kasutatav kuues kasutajaloos. Välja on toodud kaks näidet kuidas kasutuslugusid rakendada ja elektrikuludelt kokku hoida.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 39 leheküljel, 8 peatükki, 19 joonist, 1 tabelit.

Abstract

The Creation of an Electricity Market Price API Prototype

The goal of this Bachelor thesis is to create a solution that bridges the gap that is present due lack of user stories implemented by the provider electricity market price data for easy consumption in smart home ecosystems.

The solution produced by the thesis implements a vison where a dedicated application programing interface solutions transforms electricity market price to user friendly use-cases. By investigating best practices and various details requirements are gathered for the solution. In scope of the work the solution is developed, tested, and deployed to a cloud service platform. The resulting application is publicly available for consumption in Estonia for consumption of six use cases. Real world uses, both practical and theoretical are provided.

The thesis is in Estonian and contains 39 pages of text, 8 chapters, 19 figures, 1 tables.

Lühendite ja mõistete sõnastik

Access Token	*Access Token*: kasutajatuvastuse sõne
API	*Application Programming Interface*: andmepäringu rakendusliides teiste süsteemidega suhtlemiseks
CI/CD	*Continuous Integration and Continuous Delivery*: tarkvara testimise ja rakendamise tehnikate kollektsioon tagamaks tarkvaraarenduse reeglipärasus
Code Injection	*Code Injection*: ründemeetod, mis kasutab ära rakenduse sisendeid ning muudab tema käitumist
CRON	*CRON: ülesannete ajastuse programm*
CRUD	*Create, read, update, and delete*: peamised protseduurid andmetega
DEBUG	*Debug*: koodi silumise protsess, kus programmi läbitakse rida haaval
EF	*Entity Framework*: koodi ja andmebaasi suhtlusraamistik
GB, KB	*Gigabyte, Kilobyte*: digitaalse informatsiooni ühikud
GUID	*Globally Unique Identifier*: unikaalne identifikaator
HTTP GET/POST	*Hypertext Transfer Protocol GET/POT*: hüperteksti edastusprotokolli põhimeetodid, GET andmete küsimiseks ja POST andmete saatmiseks
IOT	*Internet Of Things*: asjade internet, kuhu saab ühendada igapäevaseid tarvikuid kodutehnikast autodeni
IP aadress	*Internet Protocol address*: aadress, mida võrguseadmed kasutavad teineteise leidmiseks ja identifitseerimiseks
JSON	*JavaScript Object Notation*: tekstiliste andmete struktureerimise standard
MVP	*Minimum viable product*: rakenduse minimaalne toimiv versioon
ORM	*Object–relational mapping*: tarkvaratehnika, mis võimaldab kasutada koodis andmebaasiobjekte
OWASP	*Open Web Application Security Project*: tarkvaraturvalisust edendav mittetulundusühing
PaaS	*Platform as a Service*: pilveteenuse tüüp, kus riistvara- ja platvormihaldus on automatiseeritud
PBKDF2	*Password-Based Key Derivation Function 2*: Paroolipõhine võtme tuletamise funktsioon

REST	*Representational state transfer*: veebitarkvara suhtlusstandard, mis võimaldab ligipääsu ressurssidele üle võrgu
SQL	*Structured Query Language*: päringukeel andmebaasiga suhtlemiseks
TLS	*Transport Layer Security*: võrguühenduste krüptograafiline protokoll tagamaks privaatsus, terviklus ja turvalisus
TDD	*Test Driven Development:* tarkvaraarendustehnika, kus kõigepealt tehakse testid ning alles siis kirjutatakse vastav kood.
Unix ajatempel	Number sekundites alates 1. jaanuarist 1970 Unix-i operatsioonisüsteemis
User Story	*User Story*: kirjeldus kasutaja vajadusest ning mida süsteem peab selle täitmiseks tegema
XML	*Extensible Markup Language*: tekstiliste andmete struktureerimise standard
YAML	*YAML või YML*: andmevahetuskeel, kasutatakse seadete talletamiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

2022. aasta on toonud kaasa elektri börsihinna mitmekordse kasvu ning paljudele mure, kuidas raha elektrikuludelt kokku hoida. Otsitakse võimalusi vähendada oma tarbimist – sealjuures ka elukvaliteedi arvelt. Nii mitmedki osalevad rahvakoosolekutel nõudes elektri odavamat hinda, diskuteerivad internetiavarustes odavama hinna võimalikkuse üle või otsivad tehnilisi lahendusi ning seadmeid, mis tarbivad vähem energiat. Autor on üks neist, kes piirdub lahenduste otsimisel oma koduga, püüdes muuta see piisavalt targaks, et elektrit tarbitaks kõige odavamal ajal. Targas kodus pakub kokkuhoiu võimalust nutistu [1] ehk IOT (*Internet Of Things*).

Käesolev lõputöö on tulem autori eesmärgist hoida kokku elektrihinnalt börsihinna madalpunktide kasutamise kaudu. Ühendatud on Eleringi AS elektrihinna andmed ning tark kodu. Esimeste katsetuste käigus selgus, et parema tulemuse saavutamiseks oli tarvis oodatust tehnilisemat lahendust. Probleem seisnes puuduvates kasutuslugudes, mis võimaldaks laiendada elektrihinna andmete mugavamat kasutamist. Autor näeb, et käesolev lõputöö lahendab selle probleemi ning tulemuse avalikustamine panustab laiemale kasule. Eesmärgiks on teha protüüp, minimaalse funktsionaalsusega toimiv rakendus, mis võimaldab luua uued kasutuslood.

Töö tulemusena on valminud internetis asuv avalikult kasutatav programmiliides. Selleks analüüsiti mitmekülgselt erinevaid nõudeid, loodi vajalik arhitektuur ning valiti tehnilised platvormid. Töö struktuur jälgib sarnast ülesehitust sisaldades probleemi püstitust, ülevaadet IOT maailmast ning parimatest praktikatest, nõudeid ja kaalutlusi tehnoloogiate ja majutuse valikust. Lõpus on esitatud ülevaade realisatsioonist ja tulemustest.

Rakenduse lähtekood on GitHub’is. Jooniste ja tabelite tegemiseks kasutati programme Microsoft Excel ja Visio ning olemisuhete diagrammi jaoks tarkvara Lucidchart.

Autor tänab oma juhendajaid, Meelist hea nõu ning Tiitu kannatlikust eest, sammuti ka kolleegi Erikut Azure’i nüanssides seletamise eest. Aitäh Adeliine, et mängisid natuke aega ise.

2 Probleem ja eesmärk

Alates 2021. a teisest poolest on toimunud hüppeline elektri börsihinna kasv - varasemate aastatega võrreldes (Joonis 1) on elektri hind tänaseks (sügis 2022) tõusnud kohati enam kui kümne korda [2]. Selle tulemusel tõusnud kulud panid terve Eesti rääkima ja mõtlema elektri kokkuhoiule.

Joonis 1. Elektribörsi ööpäeva hinna muutumine kuude kaupa 2020-2022.

2.1 Probleem

Tingituna kõrgest elektribörsi hinnast tekkis lõputöö autoril huvi, kuidas saaks elektri kuludelt raha kokku hoida kasutades targa kodu lahendusi. Turu-uuringute ja strateegilise konsultatsiooni pakkuja Harbor Research toob 2022. a uurimuses välja, et sarnane soov on üks peamiseid põhjuseid targa kodu lahenduse valikuks [3].

Lõputöö autor kasutab kodu automatiseerimiseks, sh tulede ja teiste IOT seadmete juhtimiseks Apple HomeKit targa kodu platvormi [4]. Viimane on osutunud autorile mugavaks lahenduseks hea ühilduvuse tõttu toetatud seadmete vahel.

Lõputöö planeerimise käigus tekkis idee ühendada Apple HomeKit targa kodu platvorm Elering AS[1] elektribörsi hinna andmetega [5]. Näiteks saaks raha kokku hoida selliste seadmetega, millel piisab töötada (näiteks öösel) mõned tunnid odava hinna perioodil (veeboiler, õhukuivati) juhtides nende tööd läbi targa kodu pistiku.

Eesti Vabariigi Riigikogu muutis seadusi [6] nii, et alates oktoobrist 2022 käivitus tarbijatele fikseeritud elektri hinnaga universaalteenus. Samas näitavad elektribörsi andmed, et hind on sageli alla määratud universaalteenuse hinna, st vähem kui 154,08€/MWh. See tähendab, et isegi universaalteenuse kontekstis säilib võimalus hoida elektri kulult kokku.

Autori katsetusete käigus selgus, et Eleringi pakutava [7] API (*Application Programming Interface*) päringu vastuses olevate andmete töötlemine erinevateks sobivateks kasutuslugudeks sobivale kujule vajab oodatust tehnilisemat lähenemist.

Probleemiks on Eleringi pakutavate andmete kuju. Päringute vastustes esitatakse on riigi hetkehind või kogu regiooni riikide ajaloolised või järgmise 24 tunni hinnad. Hinnad on megavatti kohta, ilma käibemaksuta. Kõigi ajaväljade tüübiks on Unixi ajatempel, mis edasiseks kasutamiseks vajavad konverteerimist abstraktsest numbrist kasutatavaks kuupäevaks ja ajaks. Teisisõnu, Elering tagastab päringu vastusena lihtsalt andmed ning kasutaja peab neid ise töötlema endale sobivateks kasutuslugudeks.

Järgnevas loetelus on ideed, mida autori soovib täpsustada analüüsi peatükis ning rakendada kasutuslugudena (inglise keeles *User Story*):

- Kas hind on üle või alla hinna piiri ette antud aja perioodil?
- Mitme tunni pärast, tuleb vaadeldavas ajaaknas periood, kus hind on sobiv?
- Milline on keskmine hind vaadeldavas ajaaknas?
- Kas hetke hind on väiksem universaalteenuse hinnast?
- Mis on hetke hind?

[1] Edaspidi Elering

Autor otsis, kuid ei leidnud internetist lahendusi, mis võimaldaks loetletud kasutuslugusid (leidus veebilehti kus kuvatakse hinnagraafkuid). Autor näeb seega võimalust luua lahendus, mis realiseerib eelnimetatud kasutuslood.

2.2 Lahenduse visioon

Autor näeb võimalust luua Eleringi ja targa kodu platvormi tarbijate vahele API teenus, mille eesmärk on rakendada uued kasutuslood, mida Elering ei paku, kuid mis muudaks elektri hinna andmete kasutamise lihtsamaks.

Siinse töö raames oleks realiseeritav protüübina minimaalse toimiv versioon ehk MVP[1] (*Minimum viable product*). MVP võtaks ühelt poolt sisse Eleringist tuleviku hinnad ning teisalt pakuks kõigile soovijatele lihtsa päringu vastusena juba töödeldud tulemuse – osade kasutuslugude puhul isegi jah-ei vastuse.

Sellise teenusega saaks lõputöö autor, aga ka laiem üldsus, näiteks lihtsa HTTP GET päringuga (*Hypertext Transfer Protocol GET*, päringu meetod) vastuse, mida kasutada oma targa kodu platvormil, et seeläbi juhtida seadmeid.

2.3 Eesmärgid

Probleemi lahendava visiooni elluviimiseks on lõputööl järgmised eesmärgid:

1. Analüüsida, kuidas peaks ja saaks luua API rakenduse, mis pakuks elektri börsihinda ja selle seisu läbi uute avalike kasutuslugude.

2. Valmistada rakenduse MVP vastavalt analüüsi tulemustele ning ettemääratud kasutusjuhtumitele.

3. Valideerida võimalust hoida kokku raha MVP rakenduse kasutamise tulemusena.

[1] Edaspidi kasutatakse MVP mõistet. Inglisekeelne erialane kirjandus toob välja, et nii prototüüp kui MVP näitavad lahenduse idee teostatavust. Lõputöö pealkirjaga protüüp illustreerib probleemi lahendust, kuid kirjandus eristades neid kahte MVP suutlikkusega katta kasutaja vajadust sisuliselt, midagi mida käesolev lõputöö ka sihib [55].

4. MVP raames ei ole kasutajate arv eesmärk omaette, kuid tehes rakendus internetis avalikult kättesaadavaks saab seeläbi anda võimalus selle laiemaks kasutuseks ja suuremaks kasuteguriks.

5. Jälgida MVP rakenduse majutuskulu mõistmaks rakenduse majutuse platvormi valikute täpsust ning vajadust selle muutmiseks.

2.4 Lähtetingimused ja skoop

Järgnevalt loendab autor uurimistöö suunad ja ulatuse piirangud, mis on tingitud erinevatest asjaoludest, sh autorile kättesaadavatest andmetest ja ressurssidest.

- Parimate elektrihinna jälgimise kasutuslugude leidmine vajaks laiemat kasutajakogemuse ja turu uurimist. See kahjuks ei ole mahult ega sisult võimalik antud lõputöö kontekstis. Lõputöö tulem saab olema MVP - seega kirjeldab võimalikud kasutuslood lõputöö autor, oma parima äranägemise järgi.

- Töötav MVP võimaldab päris elus katsetada suhtlust elektri hinna API ja targa kodu seadmete vahel. Sellega saab kinnitada probleemi lahendamist ning anda ka tulevikuks kaemuse lahenduse laiendamise mõistlikkusele.

- Loodav API rakendus hakkab MVP faasis kasutama laialt levinud REST (*Representational State Transfer*) suhtlusstandardit. Erinevalt teistest standarditest seisneb REST-i lihtsus selles, et ta toetab MVP kiiret loomist ja hilisemat võimalikku laiendamist [8].

- Lõputöö autor plaanib kasutada lahenduse majutamiseks tarkvaratootja Microsoft pilveplatvormi Azure teenuseid [9]. Peamine põhjus Azure teenuste kasutamiseks seisneb autori varasemas kokkupuutes ning soovis täiendada oma oskusi ja teadmisi. Teenus on tasuline, kuid autoril on olemas tasuta krediit teenuse kasutamiseks. Sellest lähtuvalt piirdub autor lahenduse loomisel andud teenusepakkuja ressursside võimalustega.

- Avalikkusele pakub elektri börsihinna andmeid Elering. Nende andmete ja API kasutamiseks on autor küsinud ning saanud emaili teel 13. septembril 2022. aastal loa. Kasutamine on tasuta ning autori on seadnud eesmärgi vältida avaliku ressurssi ülemäärast koormamist.

Tegevused, mis ei ole skoobis

- Lõputöö ei uurida elektrivõrgu ega elektribörsi toimimist, nii nende kui nutistu seadmete toimimine on mittemuudetavad välised asjaolud.

- Eesmärk ei ole võrrelda ega hinnata erinevate koduautomatiseerimise ning IOT platvormide, tehnoloogiate ja lahenduste kasutatavust või sobivust. Käesoleva lõputöö raames puudutatakse neid üksnes nii palju, et mõistma nende oskust suhelda väliste API teenustega, et kaardistada üldised nõuded loodavale API rakendusele. Ideaalis peaks ehitatav lahendus olema platvormidest sõltumatu ning kõigile avalikult ja üldiselt kasutatav.

- Targa kodu mõiste all tuntakse ka selliseid tehnoloogiad nagu Zigbee, Z-wave ja Matter. Need on võrgu kommunikatsiooni protokollid, mida reeglina targa kodu platvormid kasutavad sisemiseks andmesideks seadmetega. Kuivõrd targa kodu platvormid peidavad nad kasutaja eest tavaliselt ära, siis lõputöös neid ei uurita.

- Lahenduses ei ole tarvis tugi mitmele riigile. Eelring AS pakub Baltikumi ja Soome elektri hinna andmeid. Kuna MVP faasis ei ole oodata sealt kasutajaid, siis saab MVP toetada ainult Eestit.

- Eelring AS hinnad tulevad megavatti kohta ilma käibemaksuta. Teadmata, kuidas kasutajad lahenduse kasutusloode tulemust kasutavad ei ole otstarbekas MVP käigus rakendusel lisada käibemaksu. Hinnad võib kilovattideks teisendada.

- API dokumenteerimine laiema kasutajate ringi tarbeks on hea praktika [10]. Tehtava MVP eesmärk on probleemi lahenduse kontseptsiooni töötamise tõestamine jõudmata veel laiema kasutajaskonnani. API dokumenteerimis lahendused, nagu näiteks Swagger, jäävad peale MVP faasi selleks et toetada kasutajate liitumist.

2.5 Metoodika

Kirjeldatud on probleem ja selle taust. Sellest lähtuvalt on kujundatud lahenduse visiooni ning lähtetingimused. Kirjeldatud lähtetingimused piiritlevad skoobina nõuete leidmiseks tehtava analüüsi ulatuse ning valikud lahenduse realiseerimiseks.

Analüüsitud on nõudeid, mida lahendus peab katma ning millised tehnoloogiad on vajalikud. Nõuded sisaldavad funktsionaalseid kasutuslugusid ja mitte-funktsionaalseid tingimusi. Eritähelepanu on saanud nõuded, mis puudutavad lahenduse turvalisust ja käideldavust.

Lahenduse realiseerimiseks on tehtud vajalike raamistike valikud rakenduse programmeerimiseks, lisaks on uuritud võimalusi teenuse majutuseks ning andmebaase teabe talletamiseks. On ka oluline katta analüüsis lahenduse arendamise põhimõtteid, sisemisi osised ja üldiseid töötamise printsiipe.

Realisatsiooni käigus on kasutatud meetodeid, mis aitavad tagada arenduse kvaliteedi ja lahenduse tervikluse, näiteks lähtekoodi kaetust testidega ning regulaarset salvestamist koodihoidlas.

3 Analüüs

Käesolevas peatükis on analüüsitud ja kirjeldatud nõuded ja tehnoloogia valikud vastavalt millele, tuleb teostada lahenduse realisatsioon.

Nõuete all on käsitletud:

- MVP funktsionaalseid nõudeid lähtuvalt etteantud kasutuslugudest,
- targa kodu platvormidega suhtlemisest tingitud vajadusi,
- mittefunktsionaalseid nõudeid, sealjuures turvalisust ja käideldavust.

Tehnoloogiate valiku punktis on põhjendatud programmeerimiskeele valikut ja Azure'i rakendusplatvormi teenuste ja andmebaasi valikuid.

3.1 Nõuded

Peatükk alustab lahenduse kahe kõige olulisema osa avamisega. Esiteks, tulenevalt lõputöö lähtetingimustest kirjeldab autor MVP realiseerimiseks vajalikud kasutuslood. Selgitamaks, kuidas peaks viimaseid tehniliselt teostama, uuritakse teise sammuna viisi, kuidas targa kodu seadmed ja platvormid suhtlevad välismaailmaga.

3.1.1 Funktsionaalsed nõuded kasutuslugudena MVP jaoks

Siinses alampeatükis on API rakenduse kasutuslugude kirjeldused ja detailid. Vastavalt lähtetingimustele on nende arv ja ulatus piiratud.

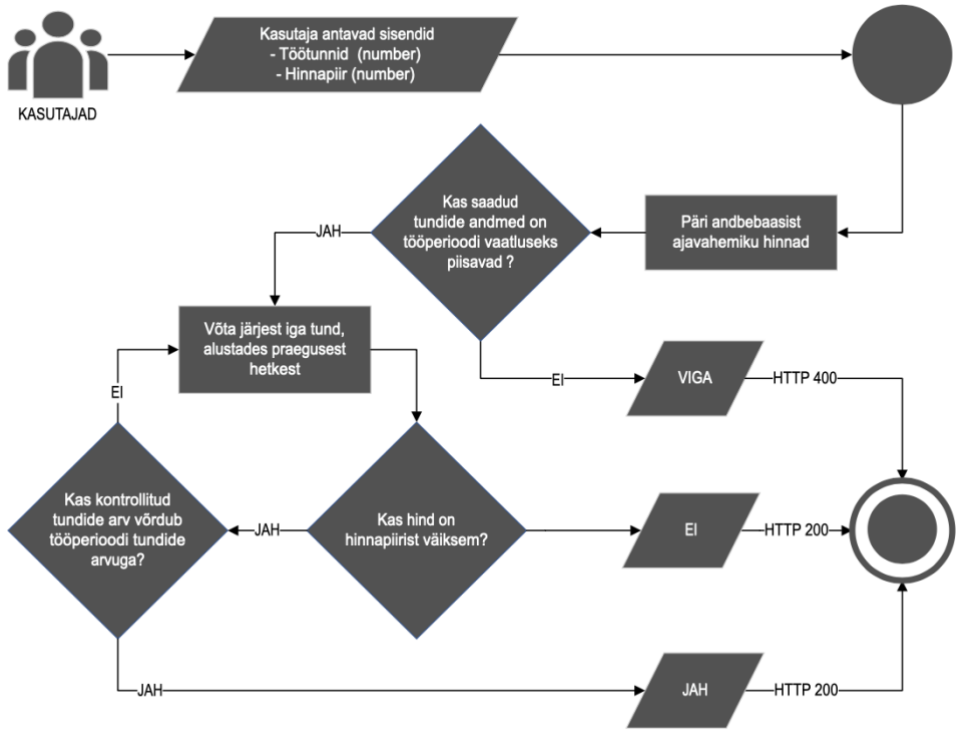
Mis on kasutuslugu? Ettevõte Atlassian, mille toode Jira on tuntud arendustööde ja projektijuhtimise töövahend, defineerib [11] kasutuslugu kui tarkvara arendusülesannet, mis kirjeldab kasutajat, tema vajadust ning eesmärki.

Lähtudes Atlassian-i määratlusest on järgnevalt sõnastatud kõik kasutuslood ja neile on lisatud ka rakenduse sisemust selgitav plokkskeem.

On oluline märkida, et kasutaja annab eesmärgi saavutamiseks kasutusloole sisendeid. Need määravad sõltuvalt kasutusloost rakenduse sisemise toimimise ja selle tulemusena kasutajale tagastatava väljundi.

Kasutuslugu 1

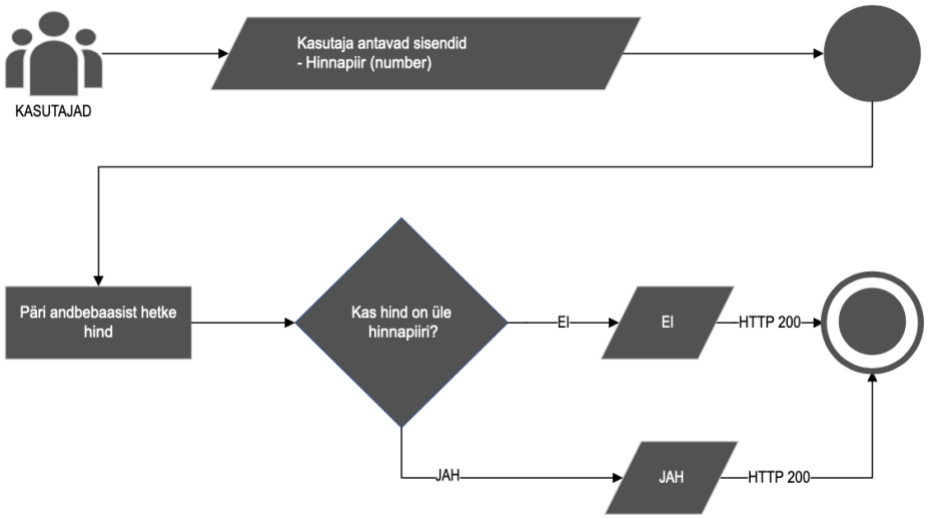
- Selleks, et panna käima mõni seade, mille töö kestvust ma tean või saan kontrollida, soovin ma saada teada, kas hetkel on periood, kus minule sobivas ajavahemikus on sobiv hind. Selleks annan ma rakendusele perioodi (arv tundides) ja hinna (number).



Joonis 2. 1. kasutusloo plokkskeem.

Kasutuslugu 2

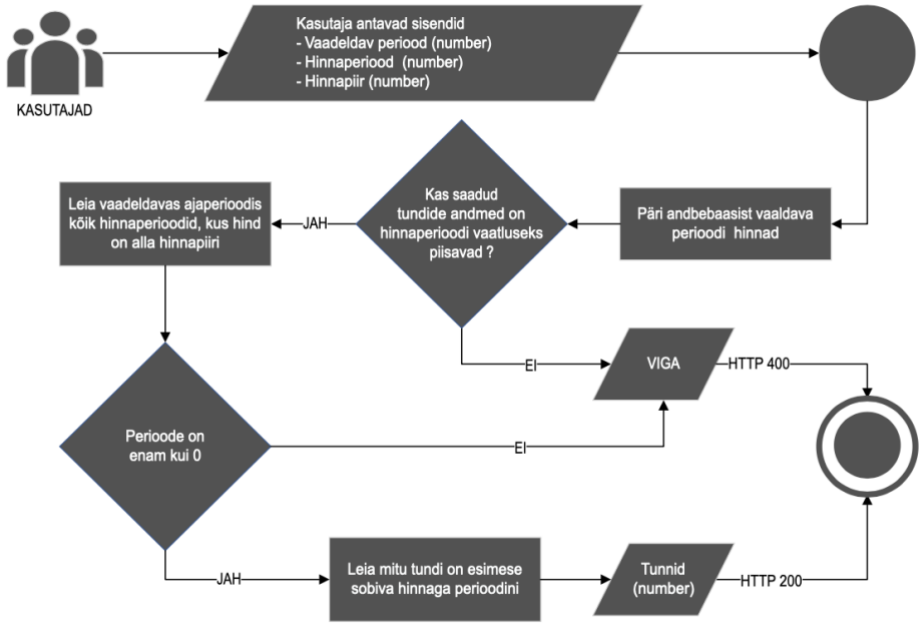
- Selleks, et ma saaks seadmed välja lülitada kui hind läheb liiga kõrgeks, soovin ma teada, kas hetke hind on ebasobiv. Selleks ma annan rakendusele hinna (number).



Joonis 3. 2. kasutusloo plokkskeem.

Kasutuslugu 3

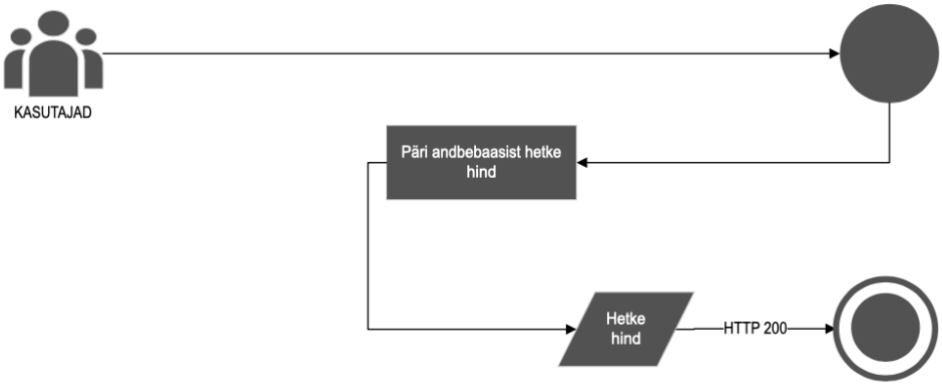
- Selleks, et planeerida enda tuleviku tegemisi, soovin ma teada mitme tunni pärast on periood, kus hind on minule sobiv teatud ajavahemikus. Selleks ma annan rakendusele vaadeldava ajavahemiku tundides (number), perioodi kestvuse (number) millal ootan sobivat hinda ja hinnapiiri (number).



Joonis 4. 3. kasutuslugu plokkskeem.

Kasutuslugu 4

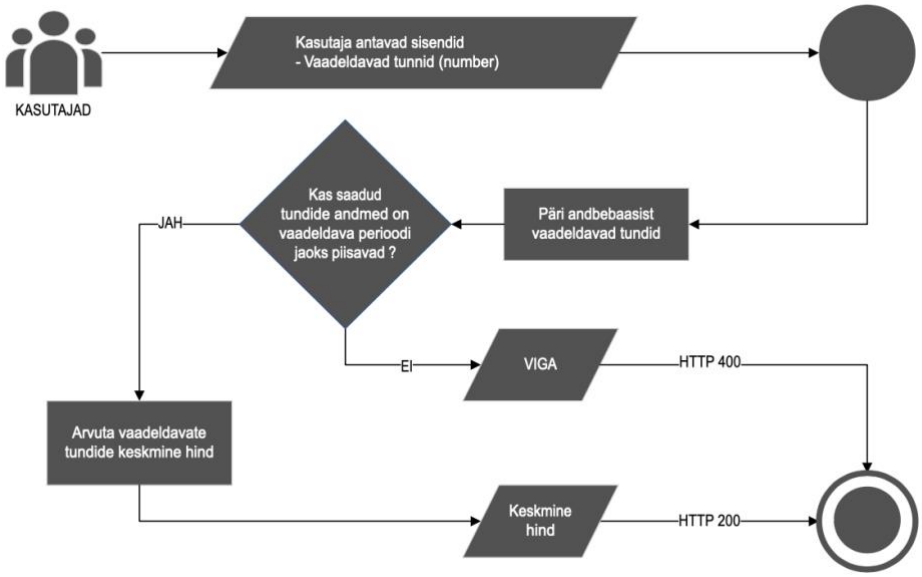
- Selleks, et ma saaks targa kodu lahenduses luua oma loogika, näiteks muuta lambipirni värvi vastavalt hinnale skaalal roheline-punane, soovin ma teada hetke hinda. Selleks ei pea ma rakendusele sisendit andma.



Joonis 5. 4. kasutusloo plokkskeem.

Kasutuslugu 5

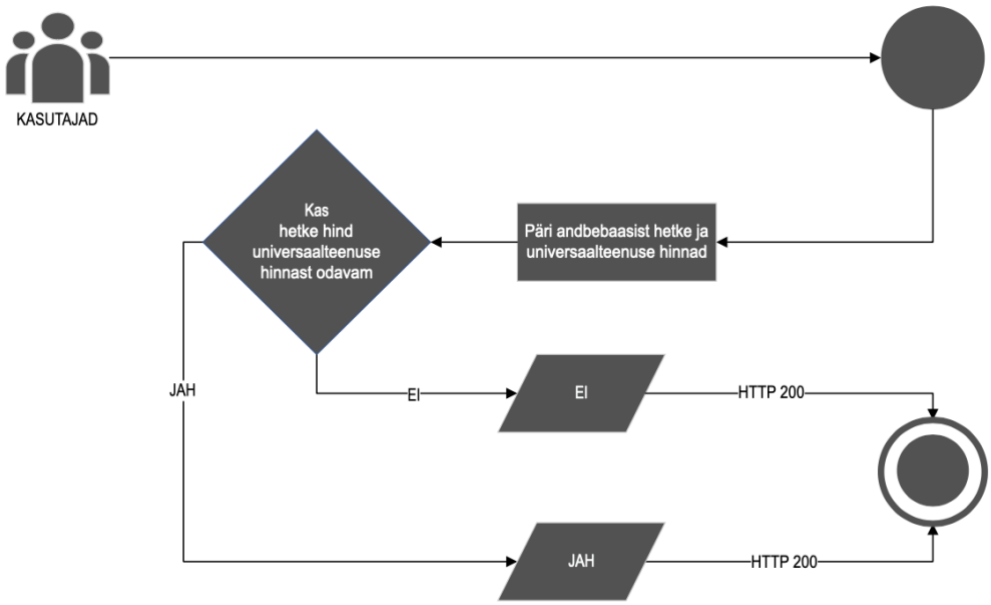
- Selleks, et ma saaks juhtida mingi seadme töö kestvust, soovin teada, mis on minu poolt etteantud tundide keskmine hind. Selleks ma annan rakendusele tundide arvu (number).



Joonis 6. 5. kasutuslugu plokkskeem.

Kasutuslugu 6

- Selleks, et minu juhitav elektritarbimine oleks odavam kui universaalteenuse kasutamine, soovin ma teada, kas hetkel on hind odavam universaalteenuse hinnast, ilma et ma teaks, mis on selle kuu universaalteenuse hind. Selleks ma ei pea rakendusele sisendit andma.



Joonis 7. 6. kasutusloo plokkskeem.

3.1.2 Nutistu ja targa kodu ülevaade

Lõputöö eesmärk on lahendada tühimik avaliku elektrihinna kasutusjuhtumite ja koduautomatiseerimist võimaldavate seadmete vahel. Selles alampeatükis määratletakse nõuded, mis tagavad oskuse nimetatud seadmetel tarbida valmiva lahenduse pakutavaid paremaid kasutuslugusid. Selleks määratleme järgnevalt võtmemõisted: IOT, selle osa targas kodus koos kasutatavate seadmete ja neid ühendavate platvormidega. Ühtlasi vaatame, kuidas need seadmed koos toimivad.

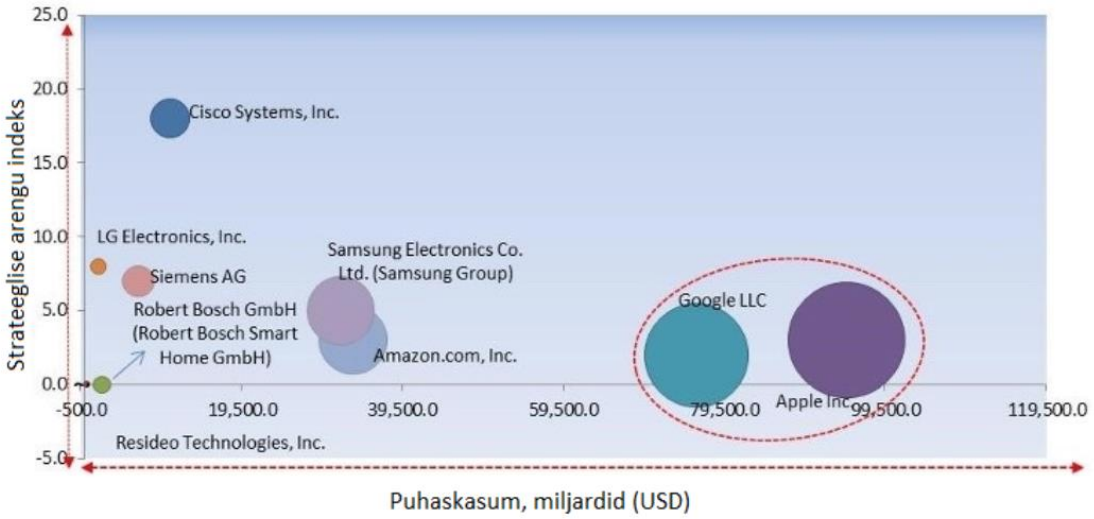
Sõnaraamat Merriam-Webster defineerib mõiste IOT kui kõiksuguste asjade ja seadmete (näiteks kodu osised, köögitehnika jne) võimet olla võrgus ning saata ja võtta vastu informatsiooni [12]. Rahvusvaheline Telekommunikatsiooni Ühing on 2012. aastal määratlenud IOT tänapäevaseks infoühiskonnaks vajaliku globaalse võrguna, mis võimaldab uuenduslike teenuseid ühendatud seadmete vahel ja neist tulevate andmete põhjal [13]. Kõigist neist võimalustest keskendub käesolev lõputöö kodule.

Juba varem nimetatud Harbor Research toob 2022. a uurimises [3] välja käsu ja kontrolli mudeli (Joonis 8), milles targa kodu platvormi keskseade on nutikõlar, mis häälkäskluste kaudu võimaldab juhtida teisi seadmeid olles seejuures ise internetiga ühenduses. Keskseadmega ühendatud teised seadmed võivad omakorda kasutada sama mudelit, laiendades võrku enda juhitud seadmetega. Sellisel viisil töötavad näiteks Philips Hue [14] ja IKEA Tradfri [15].

Joonis 8. Harbor Research-i käsu ja kontrolli mudel [3].

Ülalnimetatud mudeli juures on määrav selle keskseadet juhtiv targa kodu platvorm. Selline platvorm on siinse lõputöö tulemusel sündiva MVP rakenduse peamine kontaktpunkt ning seega nõuete formuleerimise sihtmärk nende tehniliste suhtlusomaduste osas.

Joonisel 9. on KBV Research-i välja toodud ettevõtted nende tarka kodu sissetuleku ja arengu indeksi dimensioonides. Harbor Research mudelil väljatoodud targa kodu platvormid ühtivad siin KBV Research-i loetletud targa kodu turgu domineerivate teenustega [16] ning seega sobivad need siinse töö peamiste uuritavate targa kodu platvormide hulka. Täpsemalt käsitleme järgmisi platvorme: Amazon Alexa, Google Home, Samsung SmartHome (täna Samsung SmartThings), Apple Homekit.



Joonis 9. KBV Research targakodu turg [16].

3.1.3 Nõuded rakendusele suhtlemaks targa koduga platvormidega

Järgnevalt on väljatoodud tavapärased API teenustega suhtlemise viisid, mis koos targa kodu platvormide kommunikatsiooniga moodustavad nõuded käesoleva lõputöö raames valmivale MVP lahendusele.

Lähtetingimustes on sedastatud, et API rakendus saab põhinema REST tehnoloogial kasutades suhtluseks ja andmete edastamiseks REST standardit.

- REST API võib tagastada andmeid erinevatel viisidel, neist tavapärasemad on näiteks [8]
    - Vormindamata tekstina või kuidas iganes vormindatud tekstina, näiteks XML (*Extensible Markup Language*).
    - Tavaliselt tagastab vastuse teksti kujul JSON (*JavaScript Object Notation*) vormis.
- OpenAPI spetsifikatsioonide järgi on erinevad parimad praktikad kuidas teostada kasutaja tuvastamist [17]
    - Basic, lihtne kasutaja ja parooli saatmine päringu päises.
    - API-key, unikaalne sõne kasutaja tuvastamiseks, mida edastatakse päringus.

- Bearer, krüpteeritud ligipääsu sõne, mida edastatakse päringu päises.
- OAuth 2.0, protokoll kolmanda tunnustatud osapoole kaudu ligipääsu identifikaatori saamiseks ja selle haldamiseks.

Kasutaja ei peaks saatma iga päringiga oma kasutajatunnuse ja parooli. *Bearer* ja OAuth 2.0 kasutamine ei ole vajalik MVP faasis, sest lisavad tehnilist keerukust olukorras, kus kasutajad ei edasta enda andmeid. Samuti pole vaja andmeid kasutaja kaupa eristada.

Targa kodu platvormide (Google Home [18], Amazon Alexa [19], Apple HomeKit [20], Samsung SmartThings [21]) dokumentatsioonid näitavad, et neil kõigil on võimelikus suhelda REST lahendustega ning seega peab käesoleva lõputöö tulemusel tekkiv API rakenduse MVP:

- kasutajaid registreerima (e-kirja ja salasõna alusel), mis annab uue või värske kasutajatuvastus sõne,
- kasutama kasutaja identifitseerimiseks kasutajatuvastus sõne (*API-key*),
- tagastama päringud tavatekstina või JSON-i vormingus.

3.1.4 Nõuded turvalisuse tagamiseks

Kuivõrd lõputöö tulemusel valmiv MVP saab olema avalik ning internetis kättesaadava kõigile, on oluline seada nõuded turvalisusele vältimaks halbade kavatsustega väliste osapoolte mõjusid.

Selleks kasutatakse käesolevas lõputöös tunnustatud veebirakenduste turvalisusega tegeleva organisatsiooni OWASP (*Open Web Application Security Project*) 2019. a uurimust API turvalisuse parimatest praktikatest [22]

- API3:2019 : ülemäärane andmete paljastamine – rakenduse sisemised objektid ja andmed ei ole eristatud väljaspoole edastatavatest andmetest, mis võimaldab andmete paljastust või riskantset kaemust rakenduse sisemusse. Selle vältimiseks kasutab valmiv API rakendus eraldiseisvaid klasse päringutele vastamiseks ning seega tehes sihiliku valiku näidatavatest andmetest

- API4:2019 : puudulik ressursikasutus ja päringute piiramine – rakenduse kasutamine ei ole piiratud, mille tulemusel võib tekkida tahtlikult või tahtmatult käideldavuse kadu. Selle vältimiseks peab API-rakendus registreerima kasutajad, seadma neile päringute limiidid ning reguleerima oma ressursside kasutust.

- API8:2019 : koodisüst (inglise keeles *Code Injection*), kus pahatalikult sisestatakse informatsiooni, mis võib tekitada rakenduses olukorra mida on võimalik ära kasutada. Selle vältimiseks peab API-rakendus kasutajate saadetud andmed puhastama ebavajalikust informatsioonist, kontrollima nende tüüpe ja limiite.

Teised OWASP punktid ei ole käesoleva lõputöö raames valmiva MVP jaoks määravad, kuna puudutavad funktsionaalsusi, mida ei plaanita realiseerida. Näitena võib nimetada lubamatut ligipääsu teiste kasutajate andmetele (API1:2019, API2:2019, API5:2019, API6:2019) või organisatsiooni turvalisusega laiemalt seotud protseduure (API7:2019, API9:2019, API10:2019).

Lisaks OWASP-ile on maailma üks suurim tarkvara tootmise ettevõte RedHat loetlenud järgmised API-rakenduse parimad praktikad turvalisuse tagamiseks [23]

- Kasuta ligipääsu identifikaatorit (inglise keeles *Access Token*) ehk sõne, millega tuvastada kasutajad ja kontrollida ligipääsu õigust. Alampeatükis 3.1.3 määratletakse nõue kasutajatuvastus sõne kasutamise kohta ning see eeldab ka kasutajate elementaarset registreerimist ning sõne värskendamise võimalust. Kuna need on rakendusse andmeid lisavad tegevused, siis peaks kasutama HTTP POST meetodit – see lisab turvalisust, sest HTTP POST andmeid ei hoita vahemälus ega salvestata veebilehitseja ajaloos [24].

- Krüpteeri side. Soovitud on kasutada TLS-i (*Transport Layer Security*) – krüptograafilist protokolli tagamaks võrguühenduste privaatsust, terviklust ja turvalisust. MVP faasis edastavad kasutajad oma andmeid konto loomiseks ning kasutajatuvastus sõne enda tuvastamiseks rakenduses, TLS hoiab need andmed privaatsed.

- Tuvasta haavatavused riistvaras ja tarkvaras. Lõputöö koostamisel kasutab auto rakenduse majutuseks Azure teenuseid. Viimane on PaaS (*Platform as a service*)

lahendus [25] mille pakkuja Microsoft tagab ressursside turvalisuse kindlustamise ja karastamisega [26].

- Kasuta kvoote ja piiranguid vältimaks ressursside juhuslikku või tahtlikku ülekoormust. RedHat soovitab selleks kasutada näiteks *API-key* lahendust, mis kattub varem loetletud nõuetega. See tähendab, et rakenduses peab jälgima kasutajate tehtud päringute arvu etteantud aja jooksul ning piirama päringutele vastamist piirmäära ületamisel.

- Seadistage API lüüs, mis võimaldab kontrollida võrgu liiklust. Autor leiab, et lõputöö raames API rakenduse loomiseks ei ole antud soovitus oma mahult teostatav.

3.1.5 Nõuded käideldavuse tagamiseks

Tagamaks avalikult kasutatava MVP rakenduse elementaarne käideldavus on vajalik seada nõuded käideldavuse tagamiseks. Riigi Infosüsteemide Amet defineerib käideldavuse kokkuleppena kasutuskõlblike andmete kättesaadavusest nende volitatud tarbijatele [27].

- Volitatud tarbijate haldamise vajadust oli mainitud eelmises alampeatükkis.

- Andmete, mida rakendus haldab, kasutuskõlblikuse eest vastutab lõputöö autor rakenduse loomisel. Selle parimaks tagamiseks kasutab autor TDD (*Test Driven Development*) arendusmetoodikat ning CI/CD (*Continuous Integration and Continuous Delivery*) seadistust GitHub versioonihalduses.

- Kättesaadavuse määrab suuresti lähtetingimustes kirjeldatud majutusvalik Azure pilves, kus teenusepakkuja lubab teenuste kättesaadavust 99.95% ajast [28]. Lisaks on autor leidnud võimalused reguleerida Azure teenuste kättesaadavust Azure Function abil:

    - Azure tagab ressursside piisavuse läbi sisseehitatud skaleerimise. Automaatne skaleerimine tähendab, et päringute mahu kasvades suureneb nende töötlemiseks vajalik ressurss [29]. Kuna skaleerimine töötab automaatselt, ei ole MVP raames ressursihaldusega vaja eraldi tegeleda.

- Väliste IP aadresside (*Internet Protocol* address) piiramine kui võimalus reguleerida rakenduse kasutamist ja kasutajaid võrgu tasemel [30]. See võimaldab vajadusel piirata ligipääsu rakendusele MVP-järgses faasis.

- Seadistab piirid paralleelselt töödeldavate päringute arvule, et vältida pahatahtlikku või tahtmatut ülekasutust [31].

3.1.6 Mittefunktsionaalsed nõuded

Järgnevalt kirjeldab autor mittefunktsionaalseid nõudeid, mis võimaldavad realiseerida varem kirjeldatud kasutuslugusid.

- Selleks, et pakkuda andmeid kasutajatele läbi erinevate kasutuslugude, on vaja Eleringi API-ga suheldes elektrihinna andmed tuua üle MVP rakendusse. Eleringist saadud andmed ajas enam ei muutu, andmebaasis säilitamisel on võimalik neid taaskasutada ja sellega vältida koormust, mida iga kasutaja kohta uue päringu Eleringi tegemine kaasa tooks.

- Andmete laadimine Eleringist on lahenduse töötamiseks kriitilise tähtsusega etapp. Kuna andmed puudutavad elektri tuleviku hinda, siis võimaldab see paindlikkust ebaõnnestunud päringu kordamiseks. Küsides iga tund uued andmed – perioodi kohta, mis on käesolev tund kuni 24 tundi tulevikku – on need kasutuslugude tarbeks piisavalt värsked ning üksiku laadimise ebaõnnestumist tasakaalustab järgmine päring.

- Puudub ajaloolistel andmetel põhinev kasutuslugu ja seega ei ole vaja talletada möödunud perioodide andmeid. Korra päevas toimuv andmebaasi puhastus võimaldab säästa pilveteenuse kasutamise kulusid.

- Kõik alampeatükis 3.1.1 kirjeldatud kasutuslood on realiseeritavad targa kodu seadmetele sobivalt GET päringuga, mis võimaldab anda kaasa kasutusslooga seotudparameetreid ning sealjuures määrata ka parameetritega seotudvorminguid.

- Parimad API rakenduste praktikad ütlevad, et rakenduse vastused peavad alati sisaldama HTTP protokollist lähtuvat olekukoodi [32].

3.2 Tehnoloogiate valik

Lisaks erinevatele MVP rakenduse loomise nõuetele on oluline analüüsida, milliseid tehnoloogiad ja raamistike teostusel kasutada. Käesolevas punktis on peamiseks teguriks lähtetingimustes seatud ja põhjendatud piirang Azure'i teenuste kasutamisest.

3.2.1 Rakenduse programmeerimiskeele ja raamistiku valik

MVP loomiseks on lai valik programmeerimiskeeli ja raamistikke. Arvestades lõputöö mahtu ja olemust, ei ole rakenduse autoril praktiline õppida uut keelt või raamistiku. Parem on kasutada tehnoloogiaid, mida autor tunneb ning mis võimaldavad MVP realiseerida väiksema ajakuluga, et saada kinnitus käesoleva lõputöö probleemi lahendamisele.

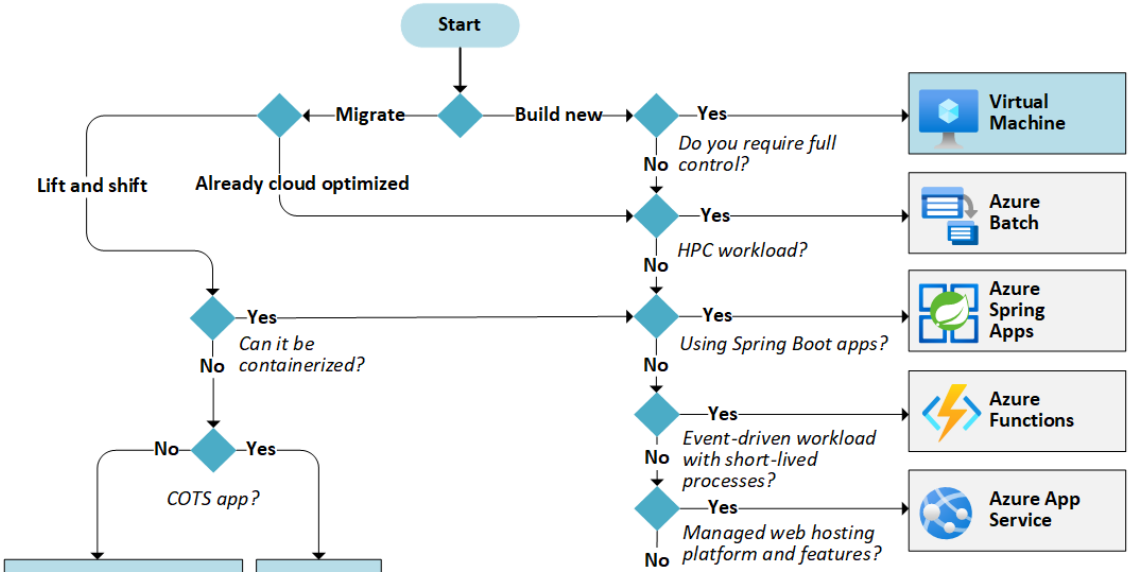
Autoril on kogemus järgnevate programmeerimiskeelte ja raamistikega (järjestatuna kogemuse järgi kahanevalt): C# (.NET), Python, Java (Spring), JavaScript (TypeScript/Node). Lähtetingimustes API rakenduse majutuseks määratud Azure'i teenused toetavad kõiki nimetatud programmeerimiskeeli.

Rakenduse realiseerimise, selle mugava testimise ja üleüldise jätkusuutlikuse huvides peab autor vajalikuks, et programmeerimiskeel oleks tugevtüübitud ja objektorienteeritud välistades Pythoni ja puhta JavaScripti.

Kuna autor ei pea oma kogemust Java (ja Spring raamistikuga) piisatavalt heaks, osutub rakenduse programmeerimiskeele valik C# ja .NET raamistiku kasuks.

3.2.2 Rakenduse majutusteenuse valik

Lähtetingimustes on seatud, et lõputöös kasutatakse Azure'i teenuseid. Otsustamaks, millist neist kasutada, sisaldub dokumentatsioonis järgnev (Joonis 10) otsustuspuu [33].



Joonis 10. Azure'i teenuste valiku otsustuspuu [33].

Uue API rakenduse loomisel on autori eesmärk lõputöö probleemi funktsionaalne lahendamine. Täiskontroll rakenduse keskkonna üle, mida pakuvad virtuaalmasinad või konteinerid, nende detailsem seadistamine ning hilisem haldamine ei anna antud juhul lisandväärtust. Samuti ei ole vajalik suure jõudlusega arvutustehnika [34], millega välistame joonisel kujutatud *Azure Batch* teenuse. Tähelepanu alt võime jätta välja ka Spring rakendused [35], sest Java kasutamine ei ole plaanis.

Sündmuspõhise *Functions* teenuse kohta selgitatakse Azure'i dokumentatsioonis, et see on lähenemine, kus arendajal ei ole vaja infrastruktuuri pärast muretseda ning ta saab keskenduda funktsionaalsusete arendamisele [36]

- Samas kirjelduses on Azure toonud välja peamised stsenaariumid, millest esimene on API rakendus ning ajatatud tegevuste käitlemine. Need sobivad täpselt käesoleva lõputöö raames tekkiva rakenduse iseloomuga: importida ajastatult andmed ning vastata kasutajate päringutele.
- Sealjuures omab *Functions* teenus punktis 3.2.1 valitud C# programmeerimiskeele tuge.

Seega saab majutusteenuseks *Azure Functions*. *Functions*'i teenustasu on kasutuspõhine ning MVP faasi tarbeks on hinnad sobivad. Hinnapoliitika ühe kuu kohta [37]

- 0,201€ - 1 miljoni päringut kohta.

- 0,000017€ - 400000GB ressursside kasutamise kohta.

3.2.3 Andmebaasi valik

Eelpool valitud C# ja .NET raamistikele pakub parimat ORM (*Object–relational mapping*) tuge Microsoft oma EF (*Entity Framework*) raamistikuga [38]. ORM vähendab SQL (*Structured Query Language*) kirjutamise ja haldamise vajadust.

EF toetab mitmeid andmebaase (SQL Server, SQLLite, Cosmos DB, PostgreSQL, MariaDB, MySQL), aga ka sellised, mida ei toeta Azure või mis on tasulised või piiratud kasutusega [39]. Nimetatud andmebaasidest ei ole Azure'i toega ainult SQLLite [40].

Pidades silmas MVP konteksti, lähtub valik Azure'i pakutavate andmebaaside kasutamise kulust. Valida on, kas ettemääratud ressursi ja mahuga pakett või reaalsel tarbimisel põhinev pakett. Täpsemalt on võimalik otsustada selle üle kas pakett on ilma määratud serverita, kuluarvestus on arvutusliku töö põhine. Sõltuvalt nendest valikutest kujuneb lõplik hind. Hea näide on Azure SQL (SQL Server) andmebaasi hinnastamine [41]. Viimasest on näha, et ressursside või mahu ettebroneerimine toob kõrgema hinna ning seega on MVP faasis parem kasutada ainult tarbimispõhist lähenemist. Selline võimalus puudub MariaDB ja PostgreSQL puhul ning seetõttu jäävad need valikust välja.

- Azure SQL (valikud *vCore* ja *serverless*) – hind on arvutusressursi kasutamise põhiselt ühe kuu kaupa, kas sekundi või tunni arvestusega, sekundi kohta 0.0001505€. Autor kalkuleeris orienteeruvaks kuluks ca 4€ kuus kui lahendus lähtub järgmistest eeldustest [41]
    - Andmebaasi maht alla 1GB (*Gigabyte*), hinnaga 0,12€/kuu
    - Iga tund kulub Eleringist andmelaadimisele 30s.
      24tundi*31päeva*30s=22320s ehk 3,35€/kuu,
    - Ühe kasutaja päringu vastamine võtab 5s.
      1kasutaja*24tundi*31päeva*5s=3720s ehk 0.56€/kuu.
- Cosmos DB (valik *serverless*) – hind kujuneb erinevate arvutusressursside kasutuse kulust päringuühiku kohta [42] ning miinimum hind on 0,26€ sisaldades ühete miljonit päringuühikut, millele lisandub 1GB andmemahtu 0,12€ eest [43]

- Kasutades asjakohast kalkulaatorit [44] on näha, et 1KB (*Kilobyte*) jagu andmete kõigi CRUD (*create, read, update, and delete*) käskude käivitamine on ca 400 päringuühikut ehk.

- Kui kasutada Azure SQL näiteandmeid, et teha ühes kuus 744 andmelaadimist ning 744 päringutele vastamist, jääb nende summa miinimumhinnas sisalduvasse mahtu.

Seega osutub valituks eeldatavasti odavaim: Cosmos DB.

Lahenduse tulemuste peatükis kajastame, milline oli andmebaasi kulu rakenduse koostamise ja selle käitlemise perioodil, et mõistma, kas on vaja muudatusi seadistustes ja ressursside valikuis.

4 Lahenduse arhitektuur

Käesolevas peatükis on kirjeldatud eelnevalt väljatoodud nõuete põhjal disaini põhimõtted, rakenduse komponendid ja sisemine struktuur aga ka andmebaasi mudel rakenduses vajalike andmete kasutamiseks.

4.1 Disaini põhimõtted

Tarkvaraarenduses on erinevaid parimaid praktikaid, millest saaks lähtuda rakenduse loomisel. Näiteks KISS (*Keep It Simple, Stupid!*) ehk kirjuta võimalikult lihtsat koodi ning DRY (*Don't repeat yourself*) ehk ära korda koodi [45].

2000-ndate aastate alguses sõnastas Robert C. Martin termini SOLID on Microsofti, mille Microsoft on nimetanud üheks kriitilisemaks tehnikaks API rakenduste teostamisel, mis seisneb järgmistes põhimõtetes [46]:

- *Single responsibility principle* – ühel asjal on üks vastutus ja ülesanne.
- *Open/closed principle* – tagatud peaks olema funktsionaalsuse laiendamise tagamine, kuid samas peaks vältima kasutuses oleva koodi katkitegemise riski.
- *Liskov substitution principle* – alamklass ei tohi oma teostuses rikkuda ülem-klassi seatud tingimusi.
- *Interface Segregation principle* – liidesed peaks olema minimaalsed, et kasutajad ei oleks sunnitud lisama neile ülemäärseid meetodeid.
- *Dependency Inversion principle* – kõrgema taseme klassid ei tohi olla otse seotud neile alluvate moodulitega, neid peavad ühendama liidesed.

Lisaks nimetatule toob Microsoft välja veel ühe parima praktika - domeenipõhise disaini (inglise keeles *Domain-Driven Design*) [47]. Domeenipõhise disaini eesmärk on tükeldada olemuslikult erinevad osad ning neist olulisemad on sõnastanud Eric Evans järgmiselt [47]

- Rakenduse kiht, mis vahendab äriloogikat suhtluses kliendiga.
- Domeeni kiht, mis keskendub andmetele ning äriloogikale olles reeglina rakenduse tuumaks. Sealjuures on ligipääs andmetele kapseldatud iga andmeklassi hoidla kaudu ehk kasutatakse hoidla mustrit (inglise keeles *Repository Pattern* [49]).

4.2 Rakenduse komponendid ja struktuur

Loodav API rakendus omab erinevaid funktsioone (näiteks andmete pärimine, kasutajapäringutele vastamine, kasutusloole vajalikud arvutused jne), mis moodustavad komponendid, millede vahel tekib rakenduse sisemine struktuur.

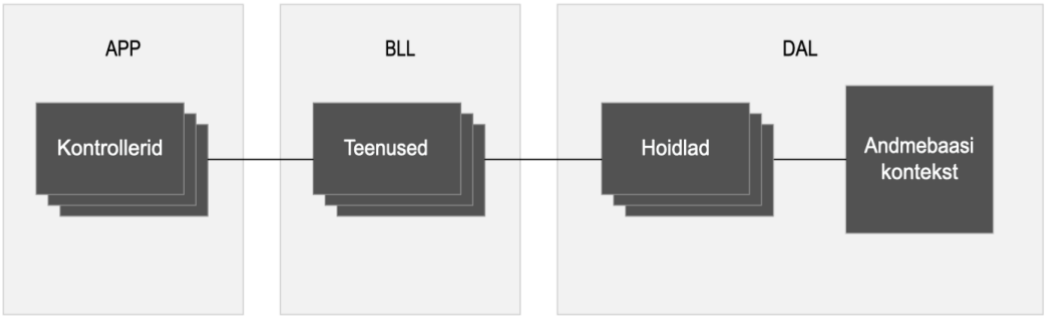
4.2.1 Rakenduse sisemised kihid ja komponendid

Käesolevas punktis on väljatoodud plaanitavad API rakenduse sisemine struktuur ja komponendid vastavalt punktis 4.1 kirjeldatud põhimõttetele.

Rakenduse sisemised eristatavad kihid on kujutatud loetelule järgneval joonisel (Joonis 11) ning need on järgmised:

- Domeeni kiht – andmete töötlemine ja ligipääsetavus rakenduses
- Kapseldatud ligipääs andmetele (inglise keeles DAL ehk Data Access Layer), mis sisaldab iga andmeklassi andmehoidlaid ning andmebaasi konteksti andmebaasiga suhtluseks.
- Äriloogika teenused, mis kasutusloo põhiselt teostavad vajalikud arvutused (inglise keeles BLL ehk Business Logic Layer). Siin formuleeritakse iga kasutusloo päringu vastus vastavalt päringule ja sellele kaasaantud parameetritele asjakohases hoidlas ning arvutatakse äriloogika kohaselt vastuse sisu.
- Rakenduse kiht, mis vastutab kliendiga suhtluse eest (inglise keeles APP ehk *Application*)
    - Iga kasutusloo API kontrollerid, mis suhtlevad vastava teenusega.

- Siin viiakse läbi kasutajatega seotud kontrollid, sealhulgas nende päringute arvu võrdlus limiitidega.



Joonis 11. Rakenduse sisemiste kihtide ülesehitus.

Koodi tasemel peavad olema eristatud järgmised klasside kogumid:

- Alusklassid ja nende liidesed.
- Andmeklasside ning andmeedastusobjektid.

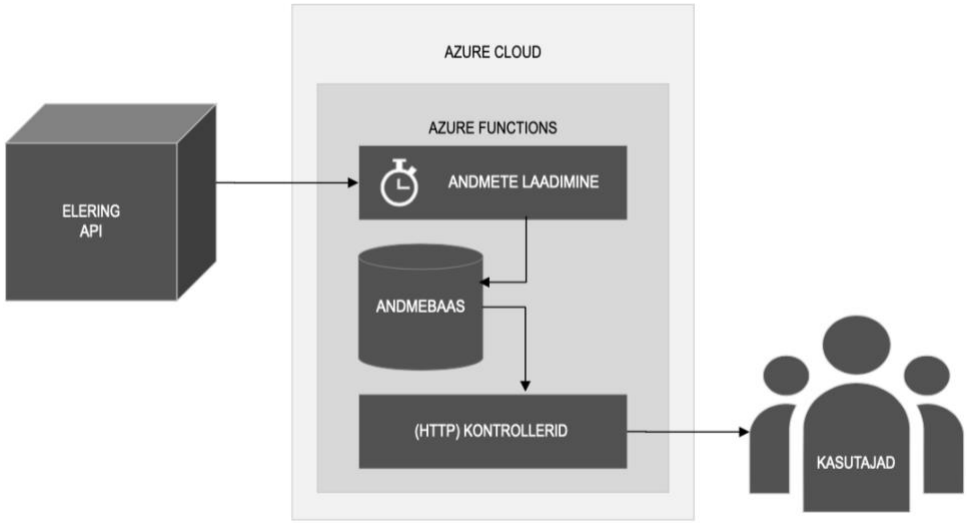
4.2.2 Rakenduse kontrollerite ja andmelaadimise käivitamine

Vastavalt kirjeldatud nõuete ja arhitektuuri peatükkidele on selles alam-peatükis kirjeldatud kuidas töötab API rakendus ning selle komponendid Azure Functions teenuses.

Functions teenuses olev rakendus töötab vastavatelt päästikutele [36]. Vajalikud päästikud sai kirjeldatud nõuete peatükis, kui väline HTTP päring ning ajastatult ehk graafiku järgi. Need päästikud kannavad sellises rakenduses kontrollerite rolli ning neid on seega vaja järgmiselt:

- HTTP päästik – iga kasutusloo kohta vastamaks kasutajate HTTP päringutele.
- Graafiku järgi – toomaks Eleringist vajalikud andmed, salvestada kasutuseks.

Järgnev joonis (Joonis 12) illustreerib kuidas kõik rakenduse kirjeldatud osised omavahel asetsevad ja opereerivad.



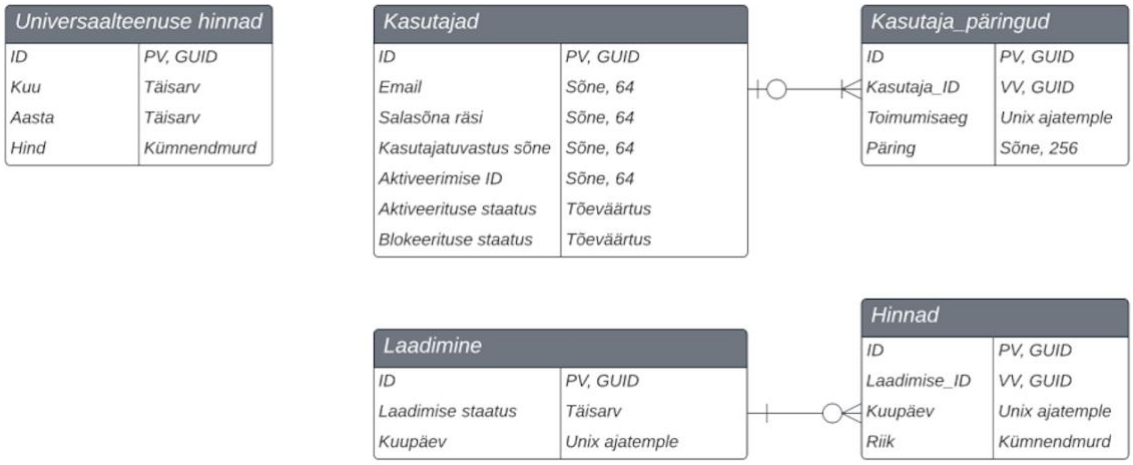
Joonis 12. Rakenduse komponendid, andmete liikumine läbi nende.

4.3 Andmebaasi olemisuhte mudel

Järgnevalt on esitletud andmebaasi olemisuhte mudel, mida MVP rakendus hakkab oma töös kasutama ja mis sisaldab iga olemi atribuute ning olemite vahelisi suhteid. Nõuete punkti 3.1.6. järgi on andmebaas vajalik, et taaskasutada andmeid ning seega vähendada Eleringi-suunalist koormust, aga ka selleks, et talletada teisi rakenduse toimimiseks vajalikke andmeid. Selliselt saab lahenduse mõlemal suunal, Eleringist andmete toomisel ja kasutajatele andmete edastamisel, lähtuda samast andmemudelist.

MVP-s ei ole palju olemeid, kasutuslugude peamine eesmärk on transformeerida Eleringist saadud andmed kasutuslugude teostuseks sobivale kujule. Kasutuslugude kirjeldus koos rakenduse nõuetega (peatükk 4) kaardistas vajadused olemite ning nende suhete kohta. Olemisuhte mudel on esitatud alloleval joonisel (Joonis 13). Täiendavad selgitused andmemudeli kohta:

- Eksisteerivad kasutajad, neid on vaja omavahel eristada; samuti tuleb jälgida nende päringute hulka.
- Toimuvad andmete sisselaadimised ning salvestatud on nende toimumised.
- Andmed, mida Elering pakub on riik, kuupäev, hinna kehtimistund, hind.
- Universaalteenuse hind iga kuu kohta.



Joonis 13. Rakenduse andmebaasi olemisuhete diagramm.

Kõikidel olemitel on primaarvõti ja seosed on läbi võõrvõtme. Võtmed on GUID tüüpi (*Globally Unique Identifier*).

5 Realisatsioon

Antud peatükis kirjeldatakse, kuidas realiseeriti MVP lahenduseks olev rakendus. Oluliste detailidena on välja toodud, kuidas teostati kasutuslood ning täideti erinevad nõuded. Sellele lisandub info evituse protsessi ja majutuse seadistustest.

Realisatsiooni teostus jaotus järgmisteks tegevuste komplektideks, mis järjestikuselt viisid lahenduse valmimise suunas:

1. Luua GitHubi versioonihalduses koodihoidla, mis on vajalik projekti lähtekoodi hoidmiseks ning rakenduse automaatseks evitamiseks Azure'i.

2. Luua Azure'i ressursid rakenduse ja andmebaasi majutuseks. Kasutada automaatset evitusprotsessi seadistust GitHubi ja *Azure Function*'i vahel. Seadistada *Azure Cosmos DB* ühendus *Azure Function* tarbeks.

3. Siduda kaks eelnimetatud Azure'is loodud komponenti, et luua rakenduse esmane kood, mis võimaldaks testida ressursside omavahelist suhtlust ning seeläbi nende seadistuste korrektsust.

4. Luua rakenduse sisemine struktuur vastavalt analüüsile ning realiseerida kasutuslood alustades andmelaadimisest. Sellele järgnevalt realiseerida kasutajate haldus, ühetaoline sisendite kontroll ja väljundite vormindamine.

5.1 Koodi teostus

Autor kasutas arendusetööriistana JetBrains Rider'it. Alternatiiv oleks olnud Microsofti Visual Studio, kuid vähene kogemus ja lõputöö formaadist tingitud ajapiirang heidutas selle kasutuselevõttu. Rider osutus mugavaks töövahendiks ja toetas hästi *Azure Function* rakenduse arendamist, viimase silumisprotsessi, kohalikku käivitamist ning algfaasis ka otse-evitust. Rider'is on lihtne alustada *Azure Function* projekti - tegemist on eraldi valikuga uute lahenduste nimekirjas.

Meetodite ja muutujate nimed ning kommentaarid on kirjutatud universaalsuse eesmärgil inglise keeles. Esialgse raamistiku loomise sammule järgnevalt on kõik sisulised meetodid loodud TDD tehnikas, et kood oleks ulatuslikult testidega kaetud. See otsus tõestas oma kasulikkust, sest võimaldas koodi ja meetodite lihtsalt ümbertegemist ning andis kindluse, et mis varem töötas, töötab ka edaspidi.

Uue projekti loomisele järgnes rakenduse sisemise struktuuri loomine vastavalt analüüsis ettenähtud põhimõttetele ja kihtidele (APP, BLL, DAL). Neile lisandus TEST kaust testide hoidmiseks ja BASE kaust, kuhu on kogutud alus-klassid. Näide rakenduse struktuurist on Lisas 2.

Illustreerimaks valminud koodi, on lisades esitatud selle Rider'i vaated koos koodi standardse värvimisega ning metaandmetega meetodite kasutuse kohta. Andmete laadimise kontrolleri *Azure Function*'i teostus on esitatud Lisas 3. Näide liideste kasutamisest olemipõhise hoidla ülemklassi näol Lisas 4. Kuivõrd hinna teenus on kasutuslugudes kesksel kohal, sisaldub selles enim meetodeid. Vaade hinna teenuse liidesele on esitatud Lisas 5.

Lõputöö esitamise hetkel on koodihoidlasse panustatud 154 korda ning koodiridu kokku 4243, millest testid moodustavad ligikaudu 48%.

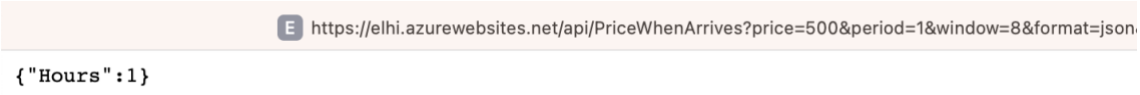
5.2 Nõuete täitmine

Järgnevalt on kirjeldatud analüüsi peatükis väljatoodud spetsiifiliste nõuete täitmine lahenduse realisatsioonis.

5.2.1 Funktsionaalsed nõuded

Rakenduses on kuus kasutusloopõhist kontrollerit, mis vastavad kasutajate GET päringutele vastavalt kasutajate antud sisenditele. Päringute vastused on vormindatud lähtuvalt päringus määratud formaadist: kas JSON või tavatekst.

Joonisel 14. on näidis 3. kasutuloo GET päringu JSON vormi vastusest.



Joonis 14. 3. kasutusloo päring ja vastus JSON vormis

Järgnev nimekiri illustreerib kõiki kasutuslugusid. Selles ei ole esitatud vormingu määratlust, rakenduse aadressi ega kasutajatuvastuse sõne (*token*). GET päringule tavapäraselt on kasutajapoolsed sisendparameetrid peale küsimärki, „parameeter=väärtus“ formaadis ning eristatud järgmisest parameetrist „&“ märgiga.

- Kasutuslugu 1: /PricePeriodUnder?price=200&period=2
- Kasutuslugu 2: /PriceOver?price=100
- Kasutuslugu 3: /PriceWhenArrives?price=200&period=1&window=6
- Kasutuslugu 4: /PriceNow
- Kasutuslugu 5: /PriceAverage?hours=1
- Kasutuslugu 6: /PriceUnderUniversal

5.2.2 Turvalisuse, mittefunktsionaalsete ja käideldavuse nõuete täitmine

Siinses alampeatükis on kirjeldatud kõik analüüsi jaotuses 3.1 väljatoodud nõuete teostuse detailid.

Kasutuslugude vastused koosnevad ühest väärtusest, kuid vältimaks andmete ülemäärase tagastamise riski, on punkti 3.1.4.a nõude jaoks, kasutusel spetsiaalsed andmeedastusklassid, mida erinevad kontrollerid taaskasutavad (ELHIPrice, ELHIHours, ELHIBoolean). Sealjuures on eraldi klass (ELHIError) päringu vea jaoks. Lisanduvad Eleringi AS API andmeklassid, mida kasutatakse sissetulevate andmete esmaseks muundamiseks (EleringData, EleringPricePoint). Lisaks on klass kasutajate registreerimiseks ja kasutajatuvastuse sõne värskendamiseks.

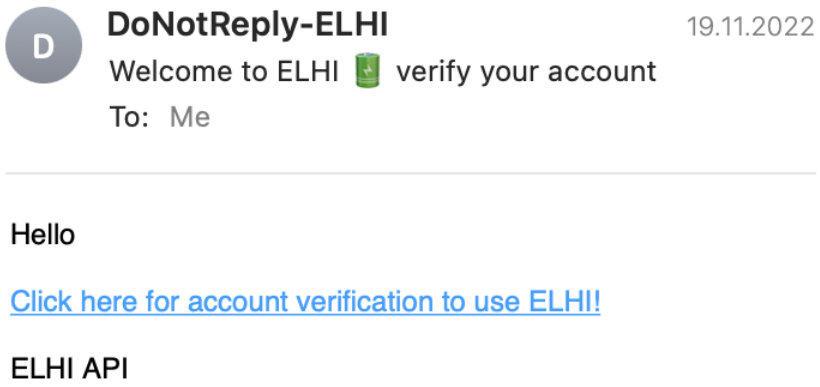
Viimane on mõeldud kasutajate ja ligipääsu haldamiseks ning täidab punktides 3.1.3 ja 3.1.4. kirjeldatud nõudeid võimaldades kasutajal:

- registreerida konto andes ette e-kirja ja salasõna,

- aktiveerida konto kinnitades oma e-kirja aadressi õigsust sellele saadetud kirja põhjal, mille sisus edastatakse kasutajale tuvastus sõne,
- uuendada tuvastussõne juhul kui see on ununenud.

POST päringu kättesaamisel kontrollitakse selles sisalduva emailina märgitud sõne vormindust ning salasõnana esitatud sõne vastavust Microsofti soovitusest lähtuvatele paroolikeerukuse nõuetele Windows 10 näitel [50]. Salasõna hoiustatakse räsina ning halduseks ja salvestamiseks kasutab rakendus .NET Core'i sissehitatud OWASP [51] PBKDF2 (*Password-Based Key Derivation Function 2*) funktsioone.

Kasutajale e-kirja saatmine kasutajakonto aktiveerimiseks on levinud praktika [52] – see on teostatud vastavalt lähtetingimustele Azure'i teenuse abil [53] – hinnaga 0,00026€ emaili kohta. Näide kasutajakonto aktiveerimise e-kirjast on esitatud alloleval joonisel (Joonis 15). Vajutades hüperlingile (sinine tekst) teeb kasutaja päringu, mis pöördub tagasi vastu rakendust ning saab vastuseks oma isikliku kasutajatuvastuse sõne.



Joonis 15. Kasutajakonto aktiveerimise e-kiri

Lisaks nimetatule kontrollib aluskontroller, kas kasutaja ei ole teinud ülemäära palju päringuid ning vajadusel blokeeritakse kasutaja edasised tegevused. Päringute piirmääraks seadis autor 100 korda viimase 24 tunni kohta ehk kasutaja saab küsida iga tund andmed ligi 3 kasutusloo kohta.

Vastavalt punktis 3.1.3. väljatoodud nõudele on vastused vormistatud kas JSON vormingus või tavatekstina. Seda funktsionaalsust pakub aluskontroller – kui päringus on lisatud vastav märge, valitakse JSON, vastaselt, ilma vormingu määratluseta saadetakse vastus tavatekstis. Aluskontrolleris asuvad meetodid, mida kasutavad kõik kontrollerid.

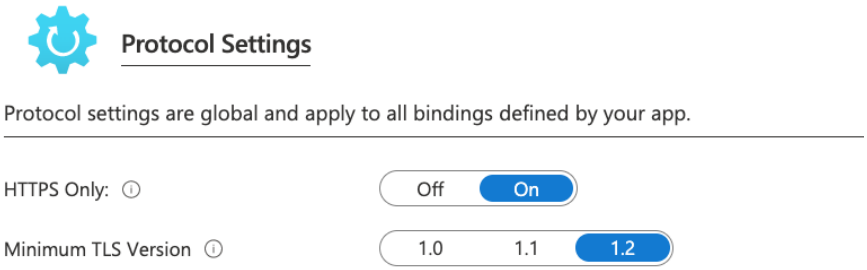
Lisaks vormindamisele on võimalik siit leida näiteks HTTP vastuse staatuse ja parameetrite koostamise meetodid, tegevuste logimine ja sisendite kontrollimine.

Sisendite kontrollimisega on täidetud punktis 3.1.4.c esitatud nõue. Kontrollitakse kõikide sisendparameetrite olemasolu, vastavust klassides nõutud tüübile, osaliselt ulatust (näiteks ajavahemiku korral püsimist viimase 24 tunni sees ja hinna puhul nullist suurema väärtuse olemasolu) ning kasutaja õigust rakendust kasutada. Kasutaja saadetud info ei lisandu otse SQL päringusse, selle välistab EF-i kasutamine.

Täiendavalt on seadistatud *Azure Function*'i päringutöötluste piirid (vaikeväärtustena 200 päringut ootel ja 100 päringut samal ajal töös) nii nagu soovitati punktis 3.1.5.

Lisaks ülalnimetatud kontrolleritele on vastavalt analüüsi punktile 3.1.6. veel kaks andmetega seotud tegevuste kontrollerit: andmete laadimiseks ja nende kustutamiseks. Laadimine toimub iga tund: Eleringilt küsitakse järgmise 24 tunni andmed. Kustutamine on seadistatud toimuma kord päevas – selle käigus puhastatakse vanad hinnad ja andmelaadimiste kirjed ning eemaldatakse mitteaktiivsed kasutajad. Kahe ajastatud kontrolleri käivitused on seadistus Azure Function CRON sõnega.

Punkt 3.1.4. soovitas seadistada *Azure Function*'is TLS'i ning see tehti Azure'i portaalis kasutades TSL'i kõrgeimat toetatud versiooni (Joonis 16).



Joonis 16. TLS seadistus

5.3 Evituse ja majutuse seadistus

Selleks, et rakenduse evitus oleks lihtne ja ilma muredeta, oli seatud analüüsis eesmärgiks rakendad automatiseeritud CI/CD protsessi, et kõik muudatused läbiks testid ja muudatus paigaldataks koheselt Azure'i.

5.3.1 CI/CD seadistus

Lõputöö esitamise hetkel on koodihoidla teinud 142 korda kooste-testi-evituse tsüklit ning kulutanud selleks 1089 minutit. GitHub Action protsessi juhib YAML (andmevahetuskeel) fail, mis on leitav Lisas 6. Selles on evitussamm *Azure Function*'i kohta, mille tulemusena uuendatakse rakendust automaatselt testide läbimisel. Siinkohal saab kiita GitHub'i, kuna see pakub tasuta kontoga 2000 tööminutit kuus, mida on oluliselt enam kui konkurentidel Bitbucket'is (50 min) ja GitLab'is (400 min).

Rakenduse testid arvutis ja GitHub'is kasutavad *in-memory* andmebaasi (andmeid hoitakse mälus, neid ei salvestata andmebaasi ega kõvakettale). Peamise rakendusevälise sõltuvuse, Cosmos DB ühenduseks vajalikud võtmed on nii kohalikus masinas kui toodangu keskkonnas eraldi seadistatud ja hoiustatud - see väldib nende juhusliku avaldamise versioonihaldusse sattumise kaudu.

5.3.2 Majutuse seadistus

Majutuseks Azure'is on kõik ressursid koondatud ühte ressursside gruppi (Joonis 17). See võimaldab neid lihtalt leida ja moodustada ühtse ülevaate kulude kohta. Kirjed *App Service Plan* ja *Application Insight* on loodud automaatselt Azure'i poolt haldamaks kasutaja tegevusi ning võimaldamaks paremat ülevaadet rakenduse haldajale. Ülejäänud ressursid on loodud Azure'i portaalis käsitsi. Ressursi loomine oli mõne hiirevajutuse tulemus, peale mida sai teostada komponentide omavahelise ühendamisega – näiteks lisades andmebaasi või e-kirja teenuse ühendussõne rakenduse keskkonna muutujatesse.

Name	Type
ASP-elhigroup-9946	App Service plan
AzureManagedDomain (elhi-email/AzureManagedDomain)	Email Communication Services Domain
elhi	Function App
elhi	Application Insights
elhi-communicationService	Communication Service
elhi-db	Azure Cosmos DB account
elhi-email	Email Communication Service
elhigroup9af9	Storage account

Joonis 17. Rakenduse ressursid Azure's

6 MVP tulemused

MVP ehitusele seati viis eesmärki. Neist kaks esimest olid ülalkirjeldatud analüüs ja sellele vastava MVP realiseerimine. Antud peatükis hinnatakse kolme ülejäänud eesmärgi täitmist.

Kõnealustest kolmest eesmärgist on kõige olulisem järgmine: *valideerida, kas MVP rakendamisel on võimalik säästa elektrikulusid*. Eesmärgi täitmise hindamiseks on autor valinud kaks meetodit: praktilise ning arvutusliku.

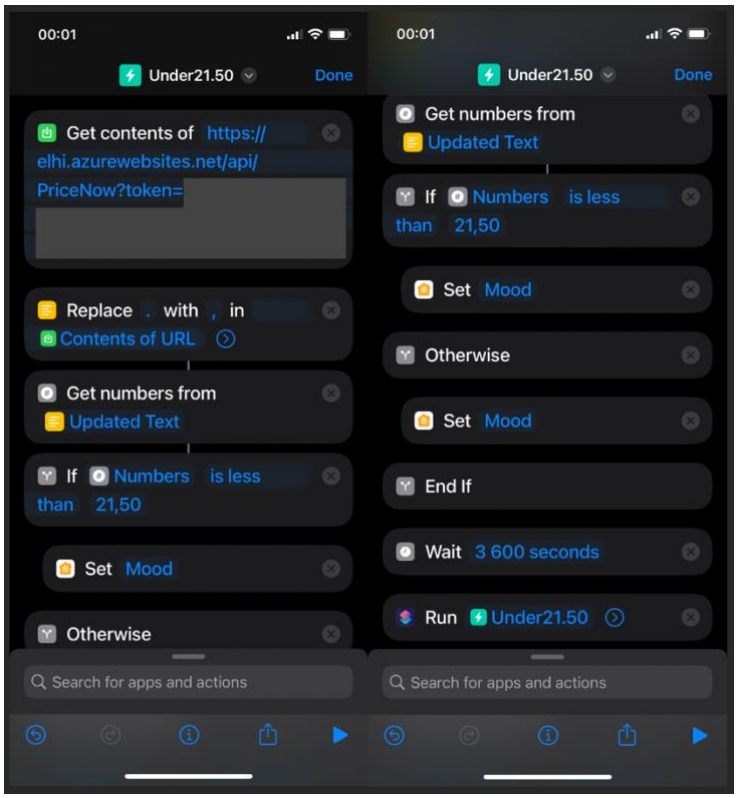
6.1 MVP elektrihinna kulude kokkuhoiu praktiline valideerimine

Kuna autor töötab peamiselt kodust, siis on suurema elektritarbimisega kodumasinate (nõudepesumasin, pesumasin-kuivati, ahi) käivitamist üritatud sättida päevase aja madalate hinnapunktide perioodidele.

See lähenemine on olnud majanduslikus mõttes mõningal määral kasulik, kuna tasud ei ole kasvanud päris samas tempos kui viimase kaheteistkümne kuu hind. Samas on selline praktika kasutajana ebamugav, nõudes pidevat käitumisdistsipliini – kõikide teiste päevaste tegevuse seas igapäevast elektrihindade jälgimist internetis ja madalpunktide saabumise aja meelespidamist.

Autori käsutuses on Apple HomeKit targa kodu lahendus. Sellega oli võimalik seadistada lambipirni värvi vastavalt etteantud programmile ning MVP eesmärk ongi pakkuda andmeid sellistele programmidele vastavalt realiseeritud kasutuslugudele.

Joonisel 18 on näha *Apple Shortcuts* programm Apple HomeKit'is kasutamiseks, milles sisalduvad erinevad sammud alustades MVP rakenduselt hetkehinna küsimisega, saadud hinna andme muutmisega tingimuslauses kasutatavaks, võrdlus autori soovitud hinnapiiriga ning vastavalt tulemusele lambipirnide värvi muutmine (kaks sammu: „*Set Mood*"). Viimane samm on oodata üks tund, et eelnevaid tegevusi korrata. Ajakulu seadistamiseks oli ligikaudu 5 minutit.



Joonis 18. Seadistus Apple HomeKit'i juhtimiseks 4. kasutusloo näitel

Selline lihtne lahendus on võimaldanud kodus olles ja väljas liikudes sujuvalt märgata sobilikku elektrihinda ning vastavalt sellele, varem valmis seatud kodutehnika käivitada. Pooleteistkuune kasutusperiood on olnud kahjuks liiga lühike, et lõputöö esitamise hetkeks tekiks selge arusaam kulude erinevuse kohta. Samas on autor üpris kindel, et antud lahendus on aidanud tarbida elektrit odava hinna perioodil ning andnud panuse igapäevastelt elektrikuludelt kokkuhoidmiseks.

Sarnane lahendus sobib väga hästi ka korteriühistus olevate elektriautode laadimise ajastamiseks – viimast on üritatud teha hinna öiste madalpunktide ajal. Elektriautode laadimiseks on koostatud eraldi seadistus, sarnaselt Joonisel 18 olevale programmile, millele on lisatud samm kasutaja telefonile teatise saatmiseks. Peale teatise kättesaamist, on kasutajal võimalik laadija kaugjuhtimise rakendusega aktiveerida. Selliselt saab mugavalt tabada ebatavaliselt odavaid hinnaperioode nagu näiteks 11. novembri ajavahemik 02:00-06:00 kus hind oli 0,01senti/kWh.

6.2 Elektrihinna kokkuhoid arvutuslikul teel

Autoril on nõukogudeaegne silikaattellistest garaaž, milles on olnud probleeme niiskusega. Siiani on aidanud keemilised niiskuseimajad, mida on olnud vaja vahetada kord kuus ja mille kulukusemäär on jäänud 6-10€ vahele. Autor on uurinud ka elektrilisi õhukuivateid, kuid nende võimalik elektrikulu on siiani heidutanud investeerimist.

Majanduslikult põhjendatud kulutuste tegemiseks põhinevad järgnevad arvutused õhukuivati alusel. Näidiseks on seade Adler AD 7917, mille ostuhind on ligilähedane keemiliste niiskuseimajate aastakuluga ning mis tarbib ühtlaselt töötades 200W elektrienergiat tunnis [54] ehk kogukulu ööpäeva kohta on 4,8lkWh.

Eesti Energia pakutava universaalteenuse hinnaga 19,24s/kWh oleks ühe seadme pideva ööpäevase töötamise kulu 0,92€. Seadme suutlikus niiskust koguda on keemilisest lahendusest oluliselt parem ehk seda ei peaks 24 tundi päevas töös hoidma, mistõttu sobib seda kasutada odavama elektrihinna perioodil. Arvestades lõputöö tegemise ajaperioodi (1. september kuni 20. november 2022) on töökulu võrdlusesse võetud järgmised tööhinna dimensioonid: börsihind, universaalteenuse hind, tööaeg perioodil kui börsihind on väiksem universaalteenuse hinnast (6. kasutuslugu) ning tööaeg kui börsihind moodustab poole universaalteenuse hinnast (1. või 2. kasutulugu).

Tabel 1. Võrdlus töökulust erinevates tööhinna dimensioonides[1]

Tööhinna dimensioonid	**Kulu kokku (eurod)**	**Töötunde perioodis kokku (tunnid)**	**Perioodi keskmine töötundide arv päevas (tunnid)**	**Osalise tööaja osakaal püsivalt töötamise ajast (protsendid)**
Börsihind kui see on pool universaalteenuse hinnast	2,14	210	2,59	10,80
Börsihind kui see on alla universaalteenuse hinna	**20,44**	**848**	**10,47**	**43,62**
Keemilise niiskuseimaja hind	15 kuni 25	-	-	-
Universaalteenuse hind	74,77	1944	24,00	-
Börsihind	91,99	1944	24,00	-

[1] Eleringi API andmete alusel, tunnihinnad 01.09.2022 00:00:01 kuni 20.11.2022 23:59:59.

Tabelis 1 on näha, et vaadeldaval perioodil oleks:

- Kõige odavam kuivati töö ainult siis, kui hind on pool universaalteenuse hinnast. Samas oleks kuivati sel juhul töötanud kõigest 10% ajast kahe ja poole kuu jooksul. Võrreldes keemiliste niiskusimajatega võib 10% tööaega olla piisav (see aspekt vajab eraldi uurimist).

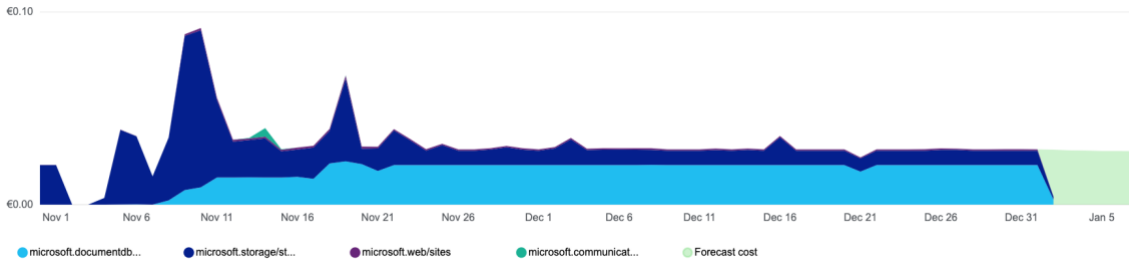
- Samas on tabelist näha, et kui seade töötaks ainult siis, kui hind on alla universaalteenuse hinna oleks kaetud 43% perioodist ning seda 3,7 kuni 4,5 korda väiksema kuluga kui börsi- või universaalteenuse hinna puhul.

Viimane on autori jaoks arvestatav tulemus, sest garaaži saaks kuivaks, töökulu (20€) oleks ligilähedane olemasolevale lahendusele (15-25€) ning vajadusel saaks kasutada vaba kuivatusvõimsust. Sellega peab autor tõestatuks nii investeeringu tegemise mõistlikkust kui ka MVP lahenduse kasutatavust.

6.3 MVP avalik kättesaadavus ja majutuskulu hinnang

Eesmärgiks oli *avalik rakendus* - see on olemas, kuid uusi kasutajaid ei ole lisandunud. Kahjuks ei ole autoril lõputöö kirjutamise kõrvalt olnud aega reklaami teha. Autor plaanib tutvustada lahendust oma kortermaja elanikele elektriautode laadimiseks ja saunakeriste soojendamiseks.

Viimaseks eesmärgiks oli hinnata *rakenduse majutuskulusid kui platvormi valiku asjakohasust ja kulu*. Lõputöö esitamise hetkeks on Joonisel 19 väljatoodud ressursid olnud kasutuses 2022. aasta novembri algusest ning selle perioodi kulud olid kokku 1,96€. Joonisel 19 on Azure portaali ekraanipilt kus on esitatud kulud ressursside lõikes koos Azure’i arvutusliku prognoosiga. Kogukulu peab autor oodatud, kahekohalisest maksumusest oluliselt väiksemaks ja seega MVP kohta heaks tulemuseks ning lahendusele edaspidi sobivaks vaikuks. Seega ei ole majutuse osas muudatusi vaja.



Joonis 19. Graafik Azure'i majutusega seotud kulude ja nende kasvuprognoosi kohta

6.4 Edasiarenduse võimalused

Rakenduse järgmine tase võiks seisneda kasutajaskonna laiendamises ning seda võimaldavates tegevustes. Sealjuures ei ole kasutajaskond piiritletud targa kodu kasutajatega – valminud API rakendusest võivad saada kasu nii koduse elektri tootjad-müüjad kui õigel ajal elektriauto laadimist soovivad autojuhid ehk kõik, kes tahavad kasutada elektribörsi hinna andmeid.

Autor jaotab võimalikud uued arendused kolmeks:

- Hetkel on kasutuslood teostatud autori parima visiooni järgi. Kuid uued kasutajad tooks uusi ideid ning need võimaldaksid omakorda kasvatada lahenduse pakutavat väärtust.
- Lisada API avalik dokumentatsioon, mis võimaldab uutel kasutajatel lihtsamalt liituda ning seeläbi tõstab rakenduse toodavat kasu.
- Hinnaülevaate koduleht võiks toimida reklaamina API kasutuslugudele. API koos veebilehega eristuks teistest elektrihinna infot pakkuvatest veebilehtedest, milledel reeglina puudub API tugi.

7 Kokkuvõte

Käesoleva bakalaureuse töö raames valmis minimaalne toimiv versioon rakendusest, mis võimaldab targa kodu lahenduste kasutajatel mugavalt tarbida elektribörsi hinna andmeid. See rakendus on realisatsioon visioonile uutest kasutuslugudest, mis Eleringi pakutavaid elektribörsi hinnaandmed ümber töödeldes on ühenduspunktiks elektritarbimise targa kodu platvormidega juhtimisel. Bakalaureusetöö alguses sai püstitatud viis eesmärki ja need said täidetud.

1. Analüüsitud on nutistu maailma targa kodu lahenduste kontekstis ja leitud nõuded, kuidas loodav rakendus peab nimetatud platvormidega suhtlema. Uurimistööga on kaetud programmiliideste loomise parimad praktikad ja välja on toodud soovitused lahenduse realisatsiooniks. Sealhulgas formuleeriti rakenduse turvalisuse ja käideldavus vajadused. Leiti printsiibid, millest juhinduda rakenduse arendamisel. Loodi nii tehnilised kui kulupõhised põhjendused tehnoloogiate ja raamistike valikuks.

2. Programmeeriti terviklik lahendus, mis toimib vastavalt analüüsis määratletud tingimustele ja kasutab selleks lähtetingimustes valitud Azure'i teenuseid. Lahendus töötab alates novembri keskpaigast, pärib Eleringilt regulaarselt uued andmed, talletab need andmebaasis ning teisaldab kujule, mida vajavad kasutajate päringutele vastamiseks kasutuslugusid esindavad rakendusliidese kontrollerid. Rakendusel on sisemiselt erinevad kihid ning komponendid, mis on ulatuslikult kaetud testidega ning võimaldavad teha koodis turvaliselt muudatusi. Rakenduse töötamise terviklust silmas pidades on seadistatud automaatne evitusprotsess, mis standardiseerib tegevusi ning kasutab nimetatud teste rakenduse töökorra kontrollimiseks enne muudatuste elluviimist.

3. Elektrihinnalt kokkuhoidmist on valideeritud kahel viisil. Autor leiab, et need kinnitavad lahenduse kasulikkust ja kasutatavust.

- Esimeseks on näide, kuidas autor juhib targa kodu seadmeid kasutades valminud rakendust. Selleks tehti targa kodu programm, mis rakenduse kasutusloost lähtuvalt muudab koduse lambipirni värvi ja seeläbi näitab kasutajale õiget aega elektri kasutamiseks.

- Teiseks arvutati koduseadme näitel, mitme töö-hinna dimensiooni võrdluses, et parim valik antud näidisseadme puhul oleks selle töötamine ainult neil tundidel, kus elektrihind on alla universaalteenuse hinna. Seeläbi saavutatakse ligi 4 korda väiksem kulu sama tulemuse kohta. Arvutuslikku näidet saab teostada targa kodu platvormides kasutades seadme juhtimiseks käesoleva lahenduse kasutuslugusid.

4. Täideti eesmärk teha rakendus internetis avalikult kättesaadavaks. Selleks on lahenduse analüüsi tulemusel, realiseerimise käigus loodud võimalused kasutajate registreerimiseks ja viisid nende ligipääsu elementaarseks haldamiseks.

5. Viimase eesmärgina tuli teha järeldusi Azure'i teenuse kasutamisest. Azure'i kasutamine sobis bakalaureusetöö kirjutamise perioodiks. Antud rakenduse näitel on kaks argumenti teiste sarnaste lahenduste loomiseks Azure'is: lihtne ja kiire seadistus ning väikesed kulud.

Lõpetuseks on autor märkinud sammud, kuidas lahendusest tulenevat kasu tõsta. Idee seisneb hetkel minimaalselt toimiva rakenduse laialdasemas kasutuses: luua programmiliidese dokumentatsiooni uute kasutajate paremaks kaasamiseks, lisada reklaamiks ja internetis kohaloluks veebilehekülg ning luua uute kasutajate vajadustest lähtudes täiendavad kasutuslood.

Kasutatud kirjandus

[1] Eesti Keele Instituut, „Sõnavõistluse võidusõna on "nuhvel",“ Eesti Keele Instituut, 4 veebruar 2018. [Võrgumaterjal]. Available: http://eurokeelehoole.eki.ee/index.php?p=1&ID=117. [Kasutatud 13 oktoober 2022].

[2] Eesti Energia Aktsiaselts, „Elektriturg - Vaatan hindade ajalugu,“ Eesti Energia Aktsiaselts, 3 november 2022. [Võrgumaterjal]. Available: https://www.energia.ee/era/elekter/elektriturg. [Kasutatud 3 november 2022].

[3] Harbor Research, „Moving Smart Home Luddites Forward,“ august 2021. [Võrgumaterjal]. Available: http://caba.org/wp-content/uploads/2022/03/IS-2022-50.pdf. [Kasutatud 10 oktoober 2022].

[4] Apple Inc, „Home App,“ Apple Inc, 15 september 2022. [Võrgumaterjal]. Available: https://www.apple.com/home-app/. [Kasutatud 15 september 2022].

[5] Elering AS, „Elektriturg,“ Elering AS, 20 oktoober 2022. [Võrgumaterjal]. Available: https://www.elering.ee/elektriturg#accordion2. [Kasutatud 20 oktoober 2022].

[6] Eesti Vabariigi Riigikogu, „Elektrituruseaduse ja konkurentsiseaduse muutmise seadus 655 SE,“ Eesti Vabariigi Riigikogu, 19 septmber 2022. [Võrgumaterjal]. Available: https://www.riigikogu.ee/tegevus/eelnoud/eelnou/c10e6791-dcda-4530-9b85-1c91b73dd997/Elektrituruseaduse ja konkurentsiseaduse muutmise seadus. [Kasutatud 26 oktoober 2022].

[7] Elering AS, „Elering dashboard API documentation,“ Elering AS, 3 september 2022. [Võrgumaterjal]. Available: https://dashboard.elering.ee/assets/api-doc.html. [Kasutatud 3 september 2022].

[8] AltexSoft, „Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC,“ AltexSoft, 29 mai 2020. [Võrgumaterjal]. Available: https://www.altexsoft.com/blog/soap-vs-rest-vs-graphql-vs-rpc/. [Kasutatud 20 oktoober 2022].

[9] Microsoft Corporation, „What is Azure?,“ Microsoft Corporation, 20 oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/. [Kasutatud 20 oktoober 2022].

[10] K. Chris, „Rest api best practices and rest endpoint design examples,“ 16 september 2021. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/. [Kasutatud 23 oktoober 2022].

[11] M. Rehkopf, „User stories with examples and a template,“ Atlasssian Corporation, 3 november 2022. [Võrgumaterjal]. Available: https://www.atlassian.com/agile/project-management/user-stories. [Kasutatud 3 november 2022].

[12] Merriam-Webster, Inc, „Internet of Things,“ Merriam-Webster, Inc, 16 oktoober 2022. [Võrgumaterjal]. Available: https://www.merriam-webster.com/dictionary/Internet%20of%20Things. [Kasutatud 16 oktoober 2022].

[13] International Telecommunication Union, „ITU-T Y.2060,“ International Telecommunication Union, juuni 2012. [Võrgumaterjal]. Available: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-E&type=items. [Kasutatud 16 oktoober 2022].

[14] Koninklijke Philips N.V., „Hue Bridge,“ Koninklijke Philips N.V., 16 oktoober 2022. [Võrgumaterjal]. Available: https://www.philips-hue.com/en-us/p/hue-bridge/046677458478. [Kasutatud 16 oktoober 2022].

[15] Inter IKEA Systems B.V., „TRÅDFRI,“ Inter IKEA Systems B.V., 13 oktoober 2022. [Võrgumaterjal]. Available: https://www.ikea.com/au/en/p/tradfri-gateway-white-40337811/. [Kasutatud 13 oktoober 2022].

[16] KBV Research, „Global Smart Home Platforms Market Size, Share & Industry Trends Analysis Report By Deployment Type (On-premise and Cloud), By Product, By Type, By Regional Outlook and Forecast, 2022 - 2028,“ KBV Research, june 2022. [Võrgumaterjal]. Available: https://www.kbvresearch.com/smart-home-platforms-market/. [Kasutatud 09 oktoober 2022].

[17] SmartBear Software, „Authentication and Authorization,“ SmartBear Software, 19 oktoober 2022. [Võrgumaterjal]. Available: https://swagger.io/docs/specification/authentication/. [Kasutatud 19 oktoober 2022].

[18] Google LLC, „Smart Home Overview,“ Google LLC, 30 märts 2022. [Võrgumaterjal]. Available: https://developers.google.com/assistant/smarthome/overview. [Kasutatud 16 oktoober 2022].

[19] Amazon Inc, „Alexa Skills Kit,“ Amazon Inc, 18 oktoober 2022. [Võrgumaterjal]. Available: https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html. [Kasutatud 18 oktoober 2022].

[20] Apple Inc, „Shortcuts User Guide,“ Apple Inc, 11 oktoober 2022. [Võrgumaterjal]. Available: https://support.apple.com/en-ca/guide/shortcuts/welcome/ios. [Kasutatud 11 oktoober 2022].

[21] SmartThings Inc, „Get Started with Cloud Connected Devices,“ SmartThings Inc, 23 oktoober 2022. [Võrgumaterjal]. Available: https://developer-preview.smartthings.com/docs/devices/cloud-connected/get-started. [Kasutatud 23 oktoober 2022].

[22] The OWASP® Foundation, mai 2019. [Võrgumaterjal]. Available: https://github.com/OWASP/API-Security/raw/master/2019/en/dist/owasp-api-security-top-10.pdf. [Kasutatud 26 oktoober 2022].

[23] Red Hat Inc, „API security,“ Red Hat Inc, 8 jaanuar 2019. [Võrgumaterjal]. Available: https://www.redhat.com/en/topics/security/api-security. [Kasutatud 17 oktoober 2022].

[24] W3schools, „HTTP Request Methods,“ W3schools, 19 november 2022. [Võrgumaterjal]. Available:

https://www.w3schools.com/tags/ref_httpmethods.asp. [Kasutatud 19 november 2022].

[25] Microsoft Corporation, „What is PaaS?,“ Microsoft Corporation, 27 oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-paas/. [Kasutatud 27 oktoober 2022].

[26] Microsoft Corporation, „Securing Azure Functions,“ Microsoft Corporation, 18 oktoober 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/security-concepts?tabs=v4. [Kasutatud 20 oktoober 2022].

[27] Riigi Infosüsteemide Amet, jaanuar 2017. [Võrgumaterjal]. Available: https://www.ria.ee/sites/default/files/content-editors/ISKE/iske_rakendusjuhend.pdf. [Kasutatud 20 oktoober 2022].

[28] Microsoft Corporation, „SLA summary for Azure services,“ Microsoft Corporation, oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/support/legal/sla/summary/. [Kasutatud 27 oktoober 2022].

[29] Microsoft Corporation, „Azure Functions hosting options / Scale,“ Microsoft Corporation, 1 september 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale. [Kasutatud 15 oktoober 2022].

[30] Microsoft Corporation, „Azure Functions networking options - Inbound networking features,“ Microsoft Corporation, 6 mai 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/functions-networking-options?tabs=azure-cli#inbound-networking-features. [Kasutatud 15 oktoober 2022].

[31] S. Lebedenko, „Azure functions. Limiting throughput and scalability of a serverless app.,“ 9 juuli 2019. [Võrgumaterjal]. Available: https://medium.com/microsoftazure/azure-functions-limiting-throughput-and-scalability-of-a-serverless-app-5b1c381491e3. [Kasutatud 15 oktoober 2022].

[32] SmartBear Software, „Best Practices in API Design,“ SmartBear Software, 17 oktoober 2022. [Võrgumaterjal]. Available: https://swagger.io/resources/articles/best-practices-in-api-design/. [Kasutatud 17 oktoober 2022].

[33] Microsoft Corporation, „Choose an Azure compute service,“ Microsoft Corporation, 27 oktoober 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/architecture/guide/technology-choices/compute-decision-tree. [Kasutatud 27 oktoober 2022].

[34] Microsoft Corporation, „High-performance computing (HPC) on Azure,“ Microsoft Corporation, 26 oktoober 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/architecture/topics/high-performance-computing. [Kasutatud 27 oktoober 2022].

[35] Micrsoft Corporation, „Azure Spring Apps,“ Micrsoft Corporation, 27 oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/products/spring-apps/. [Kasutatud 27 oktoober 2022].

[36] Microsoft Corporation, „Introduction to Azure Functions,“ Microsoft Corporation, 19 oktoober 2022. [Võrgumaterjal]. Available:

https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview. [Kasutatud 27 oktoober 2022].

[37] Microsoft Corporation, „Azure Functions pricing,“ Microsoft Corporation, 29 oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/pricing/details/functions/. [Kasutatud 29 oktoober 2022].

[38] Microsoft Corporation, „Entity Framework Core,“ Microsoft Corporation, 25 mai 2021. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/. [Kasutatud 27 oktoober 2022].

[39] Microsoft Corporation, „Database Providers,“ Microsoft Corporation, 11 mai 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli. [Kasutatud 26 oktoober 2022].

[40] Microsoft Corporation, „Types of Databases on Azure,“ Microsoft Corporation, 26 oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/products/category/databases/. [Kasutatud 26 oktoober 2022].

[41] Microsoft Corporation, „Azure SQL Database pricing,“ Microsoft Corporation, 27 oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/pricing/details/azure-sql-database/single/. [Kasutatud 27 oktoober 2022].

[42] Microsoft Corporation, „Request Units in Azure Cosmos DB,“ Microsoft Corporation, 12 oktoober 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/cosmos-db/request-units. [Kasutatud 29 oktoober 2022].

[43] Microsoft Corporation, „Azure Cosmos DB - Capacity Calculator,“ Microsoft Corporation, 29 oktoober 2022. [Võrgumaterjal]. Available: https://cosmos.azure.com/capacitycalculator/. [Kasutatud 29 oktoober 2022].

[44] A. Grant, „10 Basic Programming Principles Every Programmer Must Know,“ 20 september 2021. [Võrgumaterjal]. Available: https://www.makeuseof.com/tag/basic-programming-principles/. [Kasutatud 20 oktoober 2022].

[45] Microsoft Corporation, „Use SOLID principles and Dependency Injection,“ Microsoft Corporation, 13 aprill 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/microservice-application-layer-web-api-design. [Kasutatud 29 oktoober 2022].

[46] Microsoft Corporation, „Design a DDD-oriented microservice,“ Microsoft Corporation, 13 aprill 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice. [Kasutatud 29 oktoober 2022].

[47] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Boston: Addison-Wesley Professional, 2003.

[48] M. Fowler, Patterns of Enterprise Application Architecture, Boston: Addison-Wesley Professional, 2002.

[49] Microsoft Corporation, „Password must meet complexity requirements,“ Microsoft Corporation, 27 oktoober 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/password-must-meet-complexity-requirements. [Kasutatud 19 november 2022].

[50] OWASP, „Password Storage Cheat Sheet,“ OWASP, 14 november 2022. [Võrgumaterjal]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#pbkdf2. [Kasutatud 14 november 2022].

[51] Microsoft Corporation, „How to verify an email address in your Microsoft account,“ Microsoft Corporation, 19 november 2022. [Võrgumaterjal]. Available: https://support.microsoft.com/en-us/account-billing/how-to-verify-an-email-address-in-your-microsoft-account-c3d5eb19-ae11-fa82-1773-64f8e58e6af3. [Kasutatud 19 november 2022].

[52] Microsoft Corporation, „Azure Communication Services,“ Microsoft Corporation, 15 November 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/products/communication-services/#overview. [Kasutatud 15 November 2022].

[53] K-Rauta, „Õhukuivataja Adler AD 7917, 245 W,“ K-Rauta, 28 september 2022. [Võrgumaterjal]. Available: https://www.k-rauta.ee/p/ohukuivataja-adler-ad-7917-245-w/ifb9?cat=a77&index=7. [Kasutatud 28 september 2022].

[54] E. Merryweather, „The Difference: Prototype vs MVP,“ Product Leaders, 30 Mai 2022. [Võrgumaterjal]. Available: https://productschool.com/blog/product-management-2/difference-prototype-mvp/. [Kasutatud 26 November 2022].

[55] Microsoft Corporation, „Azure Cosmos DB pricing,“ Azure Cosmos DB pricing, 29 oktoober 2022. [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/pricing/details/cosmos-db/. [Kasutatud 29 oktoober 2022].

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

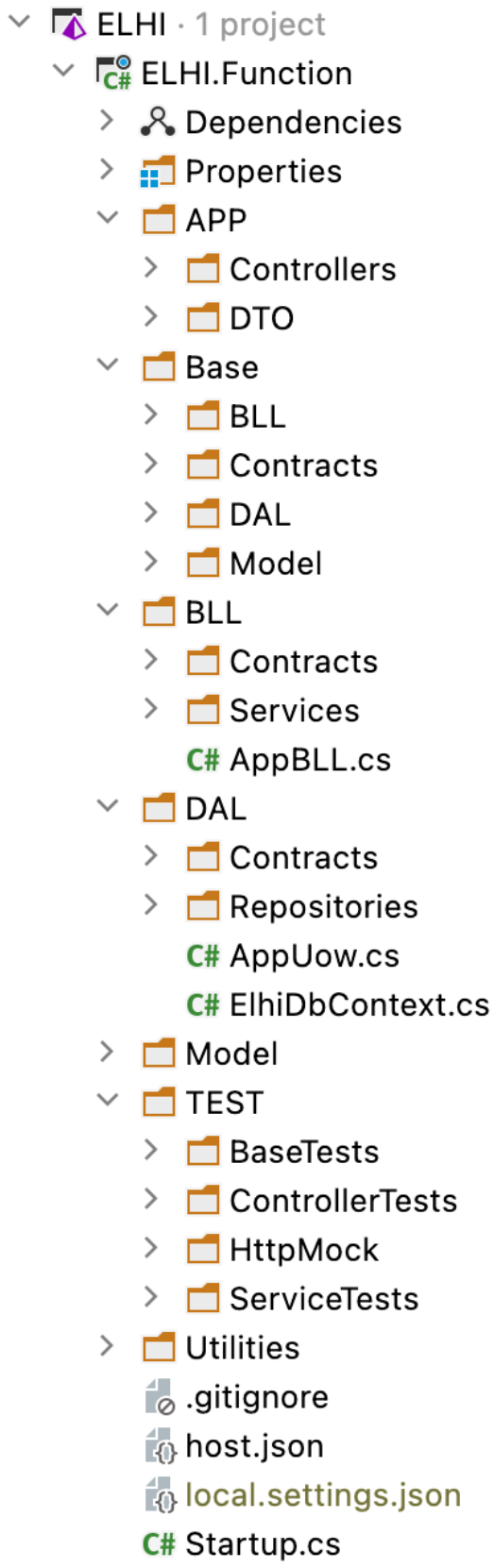
Mina, Ahto Karolin

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Elektri börsihinna API prototüübi loomine“, mille juhendajad on Meelis Antoi ja Tiit Kuuskmäe.
   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.

2.01.2023

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.

Lisa 2 – Rakenduse koodi osade struktuur



Lisa 3 – Azure Function Kontroller Rideris

```
5 usages  ahkaro *
public class PriceOver : BaseController
{
    private bool _formatJson;
    1 usage  ahkaro
    public PriceOver(IAppBll bll, ILogger<PriceOver> logger) : base(bll, logger)
    {
    }

    [FunctionName("PriceOver")]
    ahkaro *
    public async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Anonymous, params methods: "get", "", Route = null)] HttpRequest req)
    {
        _formatJson = req.Query.ContainsKey("format") && req.Query["format"] == "json";
        if (req.Query.ContainsKey("token") && ValidRequest(req.Query["token"]).Result)
        {
            await SaveUserQuery(token: req.Query["token"], query: req.Path.ToString() + req.QueryString);
            if (req.Query.ContainsKey("price") && Bll.Prices.QueryPriceIsValidNumber(req.Query["price"]))
            {
                return await ControllerLogic(req.Query["price"], logOn: true);
            }
            LogOut(logOn: true, message: "PriceOver: bad query"+ req.QueryString);
            return await Response<ElhiError>(statusCode: StatusCodes.Status400BadRequest, data: "PriceOver: bad query"
        }
        LogOut(logOn: true, message: "PriceOver: Unknown token");
        return await Response<ElhiError>(statusCode: StatusCodes.Status401Unauthorized, data: "PriceOver: Unknown toke
    }

    6 usages  ahkaro
    public async Task<ObjectResult> ControllerLogic(string price, bool logOn){...}

}
}
```

Lisa 4 – Olemipõhine hoidla ülemklass koos metaandmetega

```
namespace ELHI.Function.Base.DAL;

5 usages  5 inheritors  ahkaro
public class BaseEntityRepository<TEntity, TDbContext> : IEntityRepository<TEntity>
    where TEntity : class, IBaseEntity
    where TDbContext : ElhiDbContext
{
    private readonly TDbContext _repoDbContext;
    private readonly DbSet<TEntity> _repoDbSet;

    5 usages  ahkaro
    protected BaseEntityRepository(TDbContext dbContext){...}

    2 usages  ahkaro
    private IQueryable<TEntity> CreateQuery(bool noTracking = true){...}

    0+4 usages  ahkaro
    public TEntity Add(TEntity entity){...}

    1+1 usages  ahkaro
    public TEntity? FirstOrDefault(Guid id){...}

    0+1 usages  ahkaro
    public TEntity Update(TEntity entity){...}

    0+1 usages  ahkaro
    public bool Exists(Guid id){...}
    0+1 usages  ahkaro
    public TEntity Remove(TEntity entity){...}

    0+18 usages  ahkaro
    public IEnumerable<TEntity> GetAll(){...}
}
```

Lisa 5 – Hinna teenuse liides koos metaandmetega

```
namespace ELHI.Function.BLL.Contracts;

4 usages  1 inheritor  ahkaro  3 exposing APIs
public interface IPriceService: IEntityService<Price>
{
    5 usages  1 implementation  ahkaro
    public double? PriceNow(int timeStamp);
    3 usages  1 implementation  ahkaro
    bool AddPricesListToDb(ICollection<Price> newPriceList);
    2 usages  1 implementation  ahkaro
    ICollection<Price> FilterOutPricesThatExist(ICollection<Price> newPriceList);
    4 usages  1 implementation  ahkaro
    bool ComparePriceBesideIds(Price p1, Price p2);
    7 usages  1 implementation  ahkaro
    double  TransformQueryPriceToDouble(string queryPrice);
    15 usages  1 implementation  ahkaro
    bool QueryPriceIsValidNumber(string queryPrice);
    10 usages  1 implementation  ahkaro
    double? GetAveragePrice(int forHours);
    12 usages  1 implementation  ahkaro
    bool QueryHoursAreValid(string? hours);
    6 usages  1 implementation  ahkaro
    int TransformQueryHoursToInt(string hours);
    14 usages  1 implementation  ahkaro
    List<Price>? GetPricesBetweenStartAndLimit(int startTimeStamp, int limitTimeStamp);
    15 usages  1 implementation  ahkaro
    bool? CheckIfPriceForNowToPeriodOfHoursIsUnderLimit(double priceLimit, int periodFromNowToCheck);
    22 usages  1 implementation  ahkaro
    int? GetHoursToPeriodThatFitsGivenPricePeriodPerWindow(double priceLimit, int periodSize, List<Price> priceList);
    4 usages  1 implementation  ahkaro
    List<Price> GetOldPrices();
}
```

Lisa 6 – GitHub Action'i YAML

Workflow file for this run
.github/workflows/main_elhi.yml at 317dd3a

```yaml
# Docs for the Azure Web Apps Deploy action: https://github.com/azure/functions-action
# More GitHub Actions for Azure: https://github.com/Azure/actions

name: Build, test and deploy dotnet core app to Azure Function App - elhi

on:
  push:
    branches:
      - main
  workflow_dispatch:

env:
  AZURE_FUNCTIONAPP_PACKAGE_PATH: '.' # set this to the path to your web app project, defaults to the repository root
  DOTNET_VERSION: '6.0.x' # set this to the dotnet version to use

jobs:
  build-test-and-deploy:
    runs-on: windows-latest
    steps:
      - name: 'Checkout GitHub Action'
        uses: actions/checkout@v2

      - name: Setup DotNet ${{ env.DOTNET_VERSION }} Environment
        uses: actions/setup-dotnet@v1
        with:
          dotnet-version: ${{ env.DOTNET_VERSION }}

      - name: Build and set output for AzF package path
        shell: pwsh
        run: |
          pushd './${{ env.AZURE_FUNCTIONAPP_PACKAGE_PATH }}'
          dotnet build --configuration Release --output ./output
          popd

      - name: Run Test
        run:
          dotnet test --no-restore --verbosity normal

      - name: 'Run Azure Functions Action'
        uses: Azure/functions-action@v1
        id: fa
        with:
          app-name: 'elhi'
          slot-name: 'Production'
          package: '${{ env.AZURE_FUNCTIONAPP_PACKAGE_PATH }}/output'
          publish-profile: ${{ secrets.AZUREAPPSERVICE_PUBLISHPROFILE_331ECDEF736142C891CB80DA308580C4 }}
```