Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

Robert Leht

Mikroteenuse loomine projekti Techyon vahevarakihti

Bakalaureusetöö

Juhendaja: Helle Hein, PhD

Tartu 2023

Mikroteenuse loomine projekti Techyon vahevarakihti

Lühikokkuvõte:

Käesoleva töö raames arendatakse mikroteenus programmeerimiskeeles Go, mis hakkab kuuluma Techyoni veebiplatvormi hulka. Techyon on tarkvara teenusena (*SaaS*) platvorm, mis pakub uuenduslikke töötlus- ning analüütikavahendeid Norra elektrisüsteemi kontekstis. Arendatud teenus implementeerib Techyoni veebirakenduse hulka kuuluvat jääkaruande koostamise funktsionaalsust uuendatud kujul, mis on mõeldud parandama mahuka aruande koostamise kiirust ning stabiilsust. Jääkaruande abil saavad Norra võrguoperaatorid raporteerida oma iga-aastast analüütikat riiklikule elektrisüsteemile. Töös tutvustatakse mikroteenuseid, nendel põhinevat arhitektuuri ning teenuse arendamisel kasutatud tehnoloogiaid. Samuti selgitatakse lähemalt tarkvara ehitamise struktuuri, kus kasutajaliidese ning serveripoolsete teenuste suhtlus toimub läbi rakendusprogrammiliidese vahendaja ehk vahevarakihi.

Võtmesõnad:

Mikroteenused, Golang, vahevarakiht, jääkaruanne, Techyon

CERCS: P175 (Informaatika, süsteemiteooria)

Developing backend-for-frontend microservice for project Techyon

Abstract:

The aim of this thesis is to develop a microservice written in the programming language Go that would be deployed as part of Techyon. Techyon is a software-as-a-service platform, designed to provide innovative tools for processing and analysis within the Norwegian electrical system. The written microservice will implement a newer version of Techyon's resourceful residual report functionality, with the aim to increase its performance and stability. The residual report is a tool for Norwegian grid operators to report yearly statistics to the state. This thesis introduces microservices, microservice architecture and the technologies used for developing the freshly written microservice. The backend-for-frontend principle, also known as the API gateway structure, is introduced and explained.

Keywords:

Microservices, Golang, backend-for-frontend, residual report, Techyon

CERCS: P175 (Informatics, systems theory)

Sisukord

Sissejuhatus

Mikroteenused on viis, kuidas mugavalt rakenduse äriloogikat laiali jaotada erinevate pisemate koodihoidlate peale. Mikroteenuste eelis klassikalise monoliitse rakenduse ees seisneb mitmes aspektis. Tarkvaraprojektis, mis rakendab mikroteenuste arhitektuuri, on võimalik moodustada üksused, mis vastutavad väga konkreetse osa eest rakenduse äriloogilises pooles. Need üksused haldavad mikroteenuseid, mis vastavaid funktsionaalsuseid implementeerivad (Brown & Woolf, 2016). Tänu sellele, et kindla osa äriloogika eest vastutab sellele spetsialiseerunud meeskond, on taoliste teenuste arendus efektiivne, nende haldamine jätkusuutlik ning nende skaleerimine kasvanud/vähenenud nõudluse või andmemahu korral lihtne. Samuti on lihtne uusi funktsionaalsusi uute mikroteenuste näol hiljem sisse tuua või vananenud teenuseid kustutada.

Töö käigus arendatav mikroteenus kuulub Techyoni digilahendusse. Techyoni näol on tegemist väga mahuka digilahendusega Norra elektrisüsteemi kontekstis, mille tiimi kuulub rohkem kui 50 CGI-s töötavat spetsialisti (CGI Norge, 2023). Lõputöö autor on projektis töötanud poolteist aastat nooremarendajana ning lõputööd lugedes tuleb arvestada, et töö käigus ei laskuta liialt detailidesse projekti konfidentsiaalsuse ning domeeniteadmiste hulga tõttu.

Lõputöö kirjutamise hetkel on veel Techyoni vahevarakiht valdavas osas monoliitne. Projekti visioon on saavutada mikroteenuste arhitektuuril põhinev vahevarakiht, et vältida andmemahukate vahelehtede laadimisel kasutajaliidese ja monoliitse vahevarakihi ülekoormamist ning muuta kasutajakogemust meeldivamaks. Töö autori huvides on tutvuda lähemalt uute tarkvara ehitamise struktuuridega ning neid teadmisi rakendada projekti jaoks vajaliku teenuse arendamisel. Samuti saab töö tulemusena teada, kuivõrd paraneb aruande lehe laadimine ajaliselt, mille põhjal tuleb ka kliendi tagasiside.

Käesoleva töö eesmärk on arendada uus mikroteenus programmeerimiskeeles Go keele parimate tavade järgi, sh järgides ka Techyoni projekti käibel olevaid praktikaid. Mikroteenuse implementatsiooni eesmärk on koostada lõppkasutajale iga-aastane aruanne. Aruande funktsionaalseid nõudeid ja sisu kirjeldatakse jääkaruannete nõuete peatükis. Uus mikroteenus asendab vanema põlvkonna monoliitses koodibaasis asuvat aruande varasemat implementatsiooni. Teenuse soorituse puhul on seatud eesmärk, et arendatud teenus parandaks aruande laadimise kiirust vähemalt kümnekordselt.

Kasutatud mõisted ja terminid

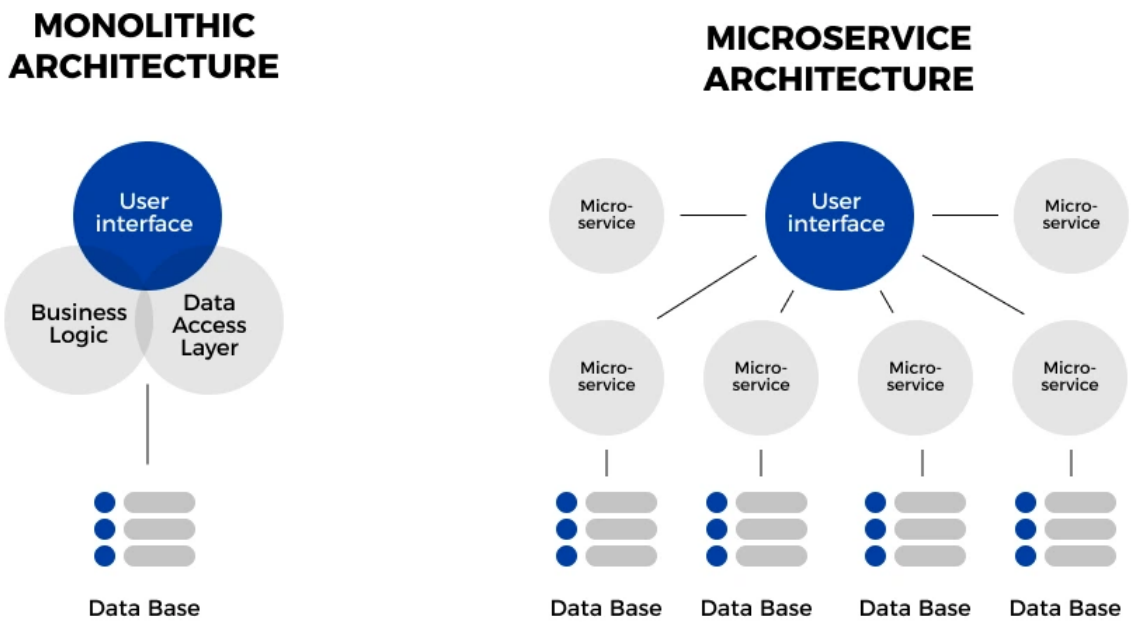
1. Ajaseeriad (ingl k *time series*) - väärtused ajas mingi kindla intervalli tagant, väärtuste ühik võib olla mõõdetud või arvutatud ning andmed tulevad tavaliselt mõõtepunktilt või seadme küljes olevalt sensorilt.
2. API (ingl k *Application Programming Interface*) - rakendusprogrammiliides, ehk komponent, mis võimaldab rakenduste vahelist suhtlust ja liidestust (*AKIT - Andmekaitse -ja infoturbe leksikon*, 28.04.2023)
3. Hind (ingl k *price*) - Statnetti poolt määratud tariif, mis on tabelis ühikuga NOK/kwH
4. Hinnatsoon (ingl k *price area*) - geograafiline ala, millel on Statnetti poolt määratud tariif
5. Kliendipoolne / kasutajaliides (ingl k *frontend*, tekstisiseselt FE või UI) - eessüsteem, läbi mille toimub kasutaja suhtlus teenusega
6. Mikroteenus (ingl k *microservice*) - spetsiifilist ja detailset äriloogikat sisaldav kergekaaluline tarkvararakendus, mis on loodud toimima paralleelselt teiste sarnaste teenustega vastavas arhitektuuris, et moodustada terviklik tarkvaralahendus
7. Munitsipaal (ingl k *municipality*) - jaotussüsteem, mille abil mõõtepunkte grupeerida
8. Mõõtepunkt (ingl k *meteringpoint*) - koht, kus mõõdetakse seda kohta läbivat elektrienergiat (*Mõõtepunkt | Energiatalgud*, 28.04.2023)
9. Serveripoolne (ingl k *backend*, tekstisiseselt BE) - kasutajale nähtamatu tagasüsteem, mis töötleb, talletab, käitleb andmeid ning loogikat (*AKIT - Andmekaitse -ja infoturbe leksikon*, 28.04.2023)
10. Tarkvara teenusena (ingl k SaaS - software-as-a-service) - tarkvara tarnimise meetod, mille puhul klient saab kaugpääsuga kasutada teenuseandja rakendusi (*AKIT - Andmekaitse -ja infoturbe leksikon*, 28.04.2023)
11. Tulemus (ingl k *result*) - Arvutatakse valemiga *keskmine* * *hind* ning kajastab lõpptulemust norra kroonides
12. Vahevarakiht (ingl k *backend-for-frontend*, ka *API gateway*) - komponent/komponentide kiht, mis vahendab päringuid klientide ja sisemiste teenuste vahel.
13. Võrguala (ingl k *grid area*) - geograafilisel alal füüsiliselt ühendatud mõõtepunktide võrgustik, mis on võimeline kandma endas mingit mõõdetavat voolu, näiteks elektrivool või vesi.

1. Teoreetiline ülevaade

Selles peatükis kirjeldatakse mikroteenuste arhitektuuri, rakendusprogrammiliidese vahendaja põhimõtet ning selgitatakse läbi Techyoni konteksti, kuidas arendada mikroteenust programmeerimiskeeles Go. Kirjeldatakse ning põhjendatakse nõudeid, mille järgi mikroteenus kirjutatakse, kuna tehnoloogilised valikud on tehtud domeeni teadmiste, projekti arhitektuuriliste reeglite ning lõppkliendi kasutusloo põhjal. Tutvutakse ka tarkvarateenusega Techyon, selle toodete ning funktsionaalsustega ja kuidas see sobiks Eesti elektrisüsteemi konteksti.

1.1 Mikroteenuste arhitektuur

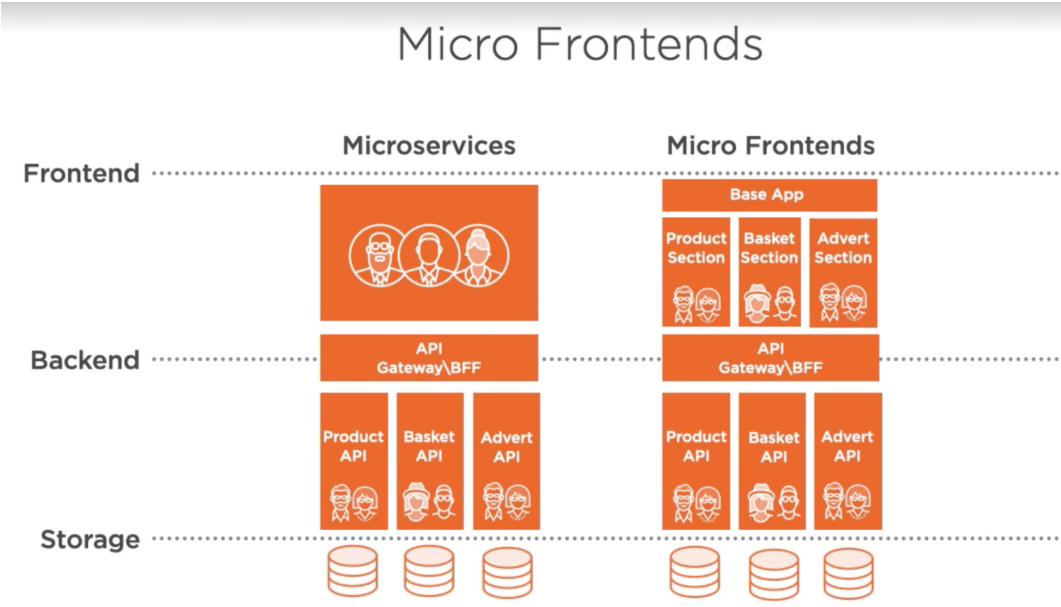
Esmalt tutvustatakse mikroteenuste arhitektuuri (edaspidi MA). Nimetatud arhitektuuri puhul moodustub üks terviklik rakendus mitmete teenuste kogumikust. Teenuste kogumikku kuuluvadki n-ö mikroteenused, mis kõik jooksevad eraldiseisvalt ning täidavad oma spetsiifilist ülesannet (Chris Richardson, 2023).



Joonis 1. Monoliitne rakendus ja mikroteenustel põhinev rakendus. (OpenLegacy, 2022)

Mikroteenuse põhimõtet saab rakendada igas tarkvara rakenduse kihis. Joonisel 1 on kujutatud võrdlus monoliitse rakenduse ning mikroteenustel põhineva rakenduse vahel. Joonise 1 mikroteenuse arhitektuuri joonisel on kujutatud rakendus, kus tagasüsteem on jaotatud

mikroteenustesse, kuid kasutajaliides on endiselt monoliitne. Samuti on joonise 1 paremal pool puudu töös käsitletav vahevarakiht, mis on see-eest monoliitselt kujutatud joonisel 2. Techyoni näitel võib eksisteerida tarkvara arhitektuur, kus mõnes kihis on näiteks üks suurem teenus, mis suure osa funktsionaalsuse eest vastutab ning kindlate protsesside skaleerimise huvides abistavad teda üksikud mikroteenused. See näitab, et töövoo ning kulude optimiseerimise huvides võib erinevate projektide puhul optimaalne arhitektuur olla täiesti erinev. Techyoni puhul on tuleviku perspektiiv mikroteenuseid rakendada igas kihis ning hetkel toimub üleminek eksisteerivatelt monoliitsetelt teenustelt täielikult ainult mikroteenustele.



Joonis 2. Monoliitne FE vs mikro FE teenused. (Sathya's Log, 2021)

MA-d rakendavas projektis suhtlevad teenused omavahel suures osas läbi API-de (ingl k *application programming interface*) (Brown & Woolf, 2016) ning teenused peavad seega teadma üksteisest ainult nii palju, et moodustada korrektne päring. Iga teenus või teenust arendav tiim vastutab ise oma API toimimise eest ning kuna ainuke kokkupuutepunkt teenuste vahel ongi API, saab projektis vajaduse või soovi korral mugavalt kasutada erinevaid tehnoloogiaid ja programmeerimiskeeli (Netflix Technology Blog, 2013). Näiteks Techyonis on kasutusel ka tagasüsteemi mikroteenused, mis on kirjutatud Kotlin programmeerimiskeeles ning lõputöö käigus osaliselt asendatav monoliitne vahevara on kirjutatud Java programmeerimiskeeles.

1.2 *Backend-for-frontend* põhimõte

Ka vahevarakihi teenuse (edaspidi ka BFF) suhtlus kasutajaliidesega toimub läbi API-de. Sellest tulenevalt võib tekkida küsimus, miks ei suhtle FE ise otse BE teenusega läbi tagasüsteemi teenuse API? Selle selgitamiseks on vaja mõista serveripoolse teenuse ning vahevarakihi teenuse põhimõtteid. Kui tagasüsteemi teenused on üldistatult mõeldud haldama kõiki taustaprotsesse ning äriloogikat rakenduses, siis vahevara teenuste eripära seisneb selles, et nad on spetsiaalselt loodud FE teenuse vajaduste täitmiseks. Vahevara teenuseid nimetatakse teisisõnu ka *API gateway*-deks, st ühe kliendipoolse päringuga koondatakse kokku kõik vajalikud päringud või andmed serveripoolsetest mikroteenustest (Chris Richardson, 2023).

Tähtis aspekt on kasutajapoolse rakenduse kiirus. Tihtipeale on vajalike andmete kuvamiseks tarvis andmeid pärida mitmest teenusest. Mitmest andmebaasist ja mikroteenusest kokku kogutud andmete kättesaamine ning töötlemine võib arvukate päringute või mahukate andmete korral võtta suure hulga aega (Netflix Technology Blog, 2013). Ilma vahevarakihi teenuseta tekib iga uue kliendi poolt tehtud päringu korral vajadus uuesti andmete pärimiseks ning töötlemiseks, mis kulmineerub aeglase kasutajakogemusega. Hästi ehitatud *API gateway* tüüpi teenuse kasutamine kasutajaliidese ning tagasüsteemi vahel võimaldab näiteks seda, et taustal valmistöödeldud andmed asuvad kergesti kättesaadaval kujul vahevarakihi andmebaasis (Microsoft, 2021).

Vahevarakiht ei ole mõeldud sisaldama äriloogikat (Brown & Woolf, 2016). Äriloogika peaks sisalduma serveripoolsetes teenustes ning BFF teenus on loodud koondama, filtreerima ja teisendama andmeid sellisele kujule, nagu seda on vaja loodavas FE komponendis. MA rakendavas projektis luuakse tihtipeale BFF ning FE komponente paarikaupa (Brown & Woolf, 2016).

Vahekiht parandab ka rakenduse turvalisust (Microsoft, 2021). Vahekiht võimaldab implementeerida valideerimist, autentimist, filtreerimist ning andmete tõlgendamist, mille abil saab vältida valede, pahatahtlike või liigsete päringute saatmist kasutajaliidesest otse serveripoolsetesse teenustesse. Sellega saab vähendada või välistada mitmeid turvariske nagu SQL-süsti, teenuste/rakenduse tahtlikku ülekoormamist, andmete lekkimist jms.

2. Techyon CGI-s

Techyon kuulub projektigruppi nimega IS Suite. IS Suite on CGI Norra intellektuaalne omand ning on (arendus-järgus) terviklik digilahendus Norra elektrisüsteemi haldamiseks. Selle eesmärgid on pakkuda haldust, analüütikat ning andmevoogude töötlust kõikidele võrgu kasutajatele, alustades elektrimüüjatest, -tootjatest ja võrguoperaatoritest kuni lõppkasutajani, ehk näiteks elumajani, mis elektrit tarbib. IS Suite'i hulka kuulub mitmeid tooteid, mõned kuni 30 aastat vanad ning suurim neist on IS Customer. Techyon on aga neist värskeim ning on loodud asendama teatud osa IS Customerist (CGI Norge, 2023). Antud lõputöös rohkem teisi tarkvaralahendusi IS Suite'i sees ei käsitleta, vaid tutvutakse pisut lähemalt vaid Techyoniga.

2.1 Techyon

Techyon on tarkvara teenusena (SaaS - *software-as-a-service*), mida müüakse erinevate toodete pakettidena. Lõputöö raames on tähtis teada, et arendatud mikroteenus kuulub *nanoFlow* toote hulka. *NanoFlow* pakub klientidele tööriistu, millega kliendi jaoks töövoogu kiirendada ning parandada. *NanoFlow* toote hulka kuulub loogika, mis raporteerib kliendile probleeme (elektrikatkestusi, potentsiaalseid vigu), kõrvalekaldeid (ebaloogilised/puuduvad andmed), ning koostab ka aruandeid, mida kliendil on vaja esitada näiteks riiklikule elektrisüsteemile või enda raamatupidamise tarbeks. Üheks selliseks aruandeks on *jääkaruanne* (ingl k *Residual report*).

2.2 Jääkaruande nõuded

Jääkaruanne kajastab andmeid, mille põhjal lõpuks koostatakse arve, mida aruande esitanu (nt elektritootja) peab maksma riigile elektrivõrgustiku kasutamise eest (Statnett, 2022). Jääkaruande otstarve on kuvada kokkuvõtlik ülevaade tootmismõõtepunktide tootmismahust munitsipaalide kaupa ning kõikide munitsipaalide tootmise summa. Kuigi lõputöö kirjutamise vältel selgub pidevalt uusi detaile aruande tegeliku praktilise kasutuse kohta, selgitatakse järgnevalt kõiki mõisteid lähemalt, mis aruande mõistmiseks vajalikud on.

Uute nõuete järgi ehitatud tabelit kujutab lisa 1. Munitsipaal on võrguoperaatori poolt loodud jaotus mõõtmispunktide otstarbeliseks grupeerimiseks. Iga tootmismõõtepunkti kohta näidatakse kümne aasta keskmist tootmist (tulp *Average*) ja iga-aastaseid tootmistulemusi (tulbad aastanumbritega), mis on kas täielikud (mitte märgitud) või poolikud (oranžina märgitud) vastavalt sellele, kas mõõtepunkt edastas andmeid terve aasta vältel. Keskmise hulka loetakse

vaid täielikud aastad. Eraldi on staatiline tulp hinna (*Price*) jaoks, mis on määratud Statnetti poolt (Statnett, 2022). Tulp nimega *Result* kajastab keskmise väärtuse ning hinna korrutist, mis on lõpliku maksu arvutamisel vajalik näitaja. Iga mõõtmispunkti kohta on ka abitulp *Alternative ID*, mis on vajalik võrguoperaatorite sisemiste protsesside jaoks. Tulp *PT [kWh]* näitab vastava mõõtepunkti kõige kõrgemat tootmist ühes tunnis, mis on kliendi poolt ette arvutatud ja määratud.

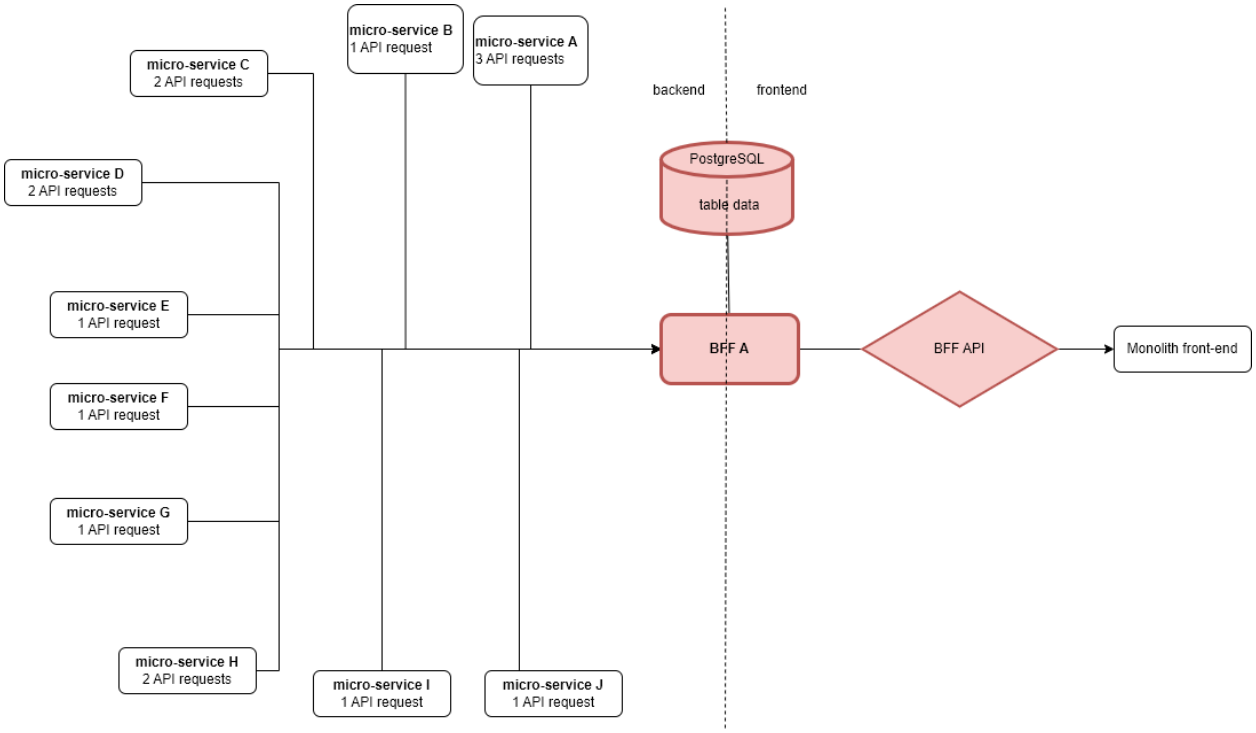
Joonis 3. Norra hinnaalad. (Statnett, 2022)

Norra pinnamood ja topograafia määravad selle, et erinevates riigi piirkondades on elektrivõrgu kasutus reguleeritud erinevate hinnaklassidega. Hinnaalade geograafiline jagunemine Norras (ning ka ümbritsevates riikides) on kujutatud joonisel 3. Tabelis on ka tulbad, mis näitavad mõõtmispunktide kuuluvust elektrivõrku (*Grid Area*) ja hinnaklassi (*Price Area*) nagu lisas 1 ning neid parameetreid määrab riik geograafilise asukoha alusel. Jääkaruanne proovib pakkuda viisi, kuidas kõiki vajalikke andmeid hoomataval moel kliendile kuvada ning vormistada. Mittefunktsionaalse nõudena tuli kliendi poolt soov, et aruande laadimine oleks võimalikult kiire, eesmärgiks seati 2 sekundit.

2.3 Vajadus uue mikroteenuse kirjutamiseks Techyoni vahevarakihti

Varem baseerus aruandluse funktsionaalsuse implementatsioon monoliitses BFF koodihoidlas. Vana BFF puhul ei osatud veel projekti UI komplektsust hinnata, mistõttu puudub sel andmebaas. See tähendab seda, et vana BFF ei töödelnud andmeid taustal, ega laadinud

midagi ette, vaid pidi iga UI-st tuleva päringu peale hakkama BE-st andmeid koguma ning töötlema. Samuti lisandus ajapikku sinna niivõrd palju funktsionaalsusi, et teenuse jõudlus hakkas selle tõttu langema. Paljud ressursimahukad päringud tekitasid olukorra, kus kasutajaliides muutus kasutamatuks seni, kuni andmed BE-st täielikult sisse laeti. Vana BFF puhul esines pikkade päringute puhul ka probleem, et nende täitmisele kulus rohkem kui 30 sekundit - UI-s on implementeeritud automaatne *time-out*, kui andmete kättesaamine võtab aega kauem kui pool minutit.



Joonis 4. Arendatud teenuse paiknemine tagasüsteemi ja kasutajaliidese vahel.

Ka samaaegsed päringud ning korduvad optimiseerimised ei parandanud seda olukorda jääkaruande funktsiooni puhul. Seega oli vaja ehitada uus BFF teenus, mis oleks võimeline andmed kõigist vajalikest mikroteenustest koguma taustal, neid optimaalsel kujul hoiustama ning ühe API päringu kaudu UI-le edastama. Uue mikroteenuse paiknemine projekti kontekstis on kujutatud joonisel 4.

3. Metoodika

Selles peatükis kirjeldatakse lähemalt mikroteenuse ehitamiseks valitud tehnoloogiaid ning arendusprotsessi. Töö praktilise osa ehitamiseks kasutatakse rakendust IntelliJ IDEA. Mikroteenuse kirjutamisel kasutatakse programmeerimiskeelt Go, andmete haldamiseks kasutatakse PostgreSQL andmebaasi haldamise süsteemi ning mikroteenuse töö tulemuse näitamiseks mõeldud kasutajaliidese loomiseks Javascripti raamistikku Vue. Järgnevalt kirjeldatakse mikroteenuse kirjutamiseks valitud tehnoloogiaid.

3.1 Mikroteenuse ehitamisel tehtud tehnoloogilised valikud

Go on Google'i poolt välja arendatud protseduuriline programmeerimiskeel. Aastast 2009 on ta tugev konkurent Java keelele mikroteenuste kirjutamisel, kuna on ehitatud pidades silmas mitmelõimelisust ning skaleeritavust (Andrawos & Helmich, 2017). Techyoni kontekstis on Go valitud peamiseks *back-end* programmeerimiskeeleks mitmel põhjusel. Esiteks on Go kergesti õpitav tänu oma lihtsale süntaksile ning intuitiivsetele standardteekidele. Go on väga hästi ühildatav Google'i poolt pakutava Google Cloud platvormiga, mille peal jooksevad kõik Techyoni teenused. Samuti on Go rakenduse käivitamisaeg märgatavalt kiirem vanadest Java Spring Boot rakendustest, kuna kompileerub otse masinkoodiks erinevalt Javast, mis kasutab kompileerimiseks virtuaalmasinat.

Andmebaasi haldamiseks on kasutusel PostgreSQL. PostgreSQL (või Postgres) on avatud lähtekoodiga relatsioonilise andmebaasi haldussüsteem. Valitud süsteemi kasutatakse töö kirjutamise hetkel enamikes Techyoni mikroteenustes ning on väikese kuni keskmise mahuga andmete puhul projektis standardiks.

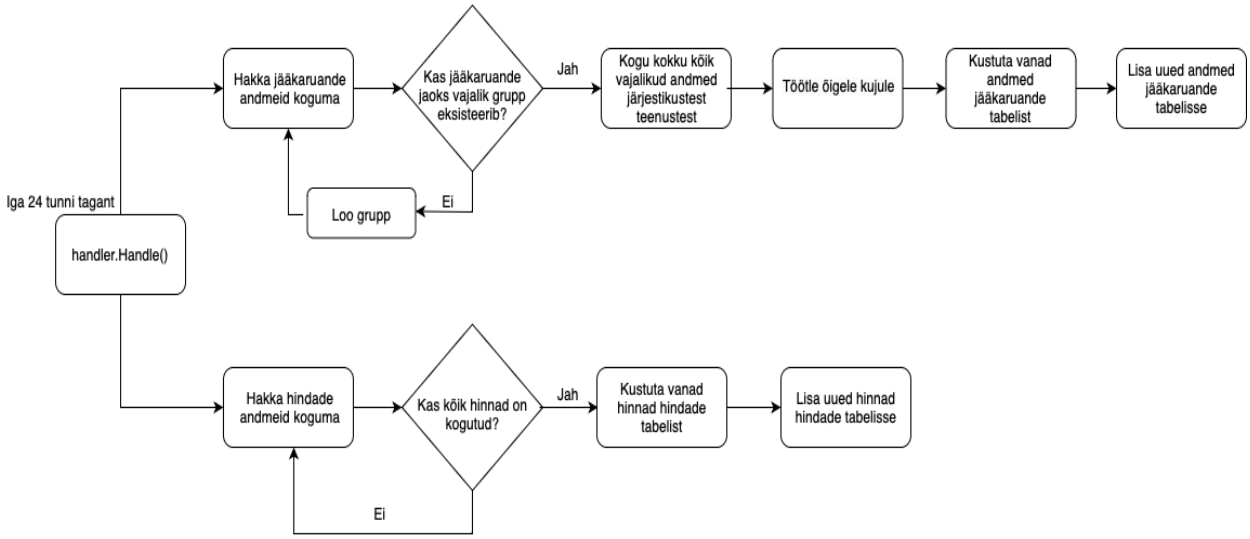
Mikroteenuse poolt genereeritud aruannet kuvatakse projekti veebirakenduse kasutajaliideses (edaspidi UI - *user interface*). UI loomiseks on kasutatud FE raamistikku Vue. Vue on populaarne JavaScripti raamistik, mis on tuntud oma väikese mahu ning kergesti õpitavuse poolest (Kofi Group, 2021). Otsuse kasutada Vue raamistikku määras projekti arhitektuur, nimelt on projekti hetkel monoliitne FE koodibaas kirjutatud Vue raamistikus.

3.2 Teenuse kirjutamise protsess

Techyoni poolne initsiatiiv kirjutada täiesti uus teenus jääkaruande funktsionaalsuse jaoks tulenes monoliitse vahekihi kehvast jõudlusest ning selle ülekoormuse vähendamisest.

Monoliitse vahekihi implementatsiooniga võttis aruande koostamine aega ca 10 sekundit ning muutis ülejäänud veebirakenduse kliendi jaoks kasutamatuks. Selline jõudlus ei olnud Techyoni tiimi jaoks aktsepteeritav kasutajakogemuse ega veebirakenduse toimivuse vaatenurgast.

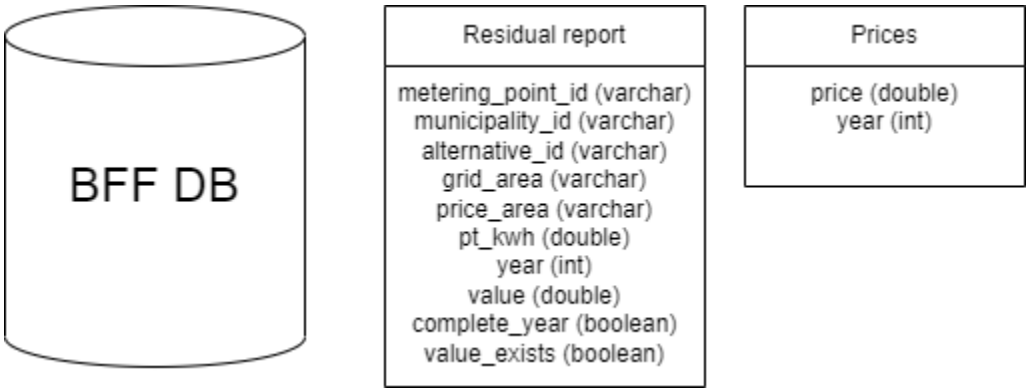
Esimene faas oli luua uus mikroteenus ning viia selle sisu sellisesse seisu, kus toimuks kogutud andmetega sama töötlus, mis toimus vanas monoliitses vahevara teenuses. Sellele lisandus etapp lisada need töödeldud andmed vastloodud andmebaasi. Esimeses faasis kuulus andmebaasi kolm tabelit. Nullist kirjutamise protsessi käigus sai uuesti kogu loogikavoo üle analüüsida ning kohandada selle ümber Go programmeerimiskeele ning mikroteenuste eripäradele.



Joonis 5. Lihtsustatud kuju lõplikust andmete kogumise voost.

Uue teenuse kirjutamise teises faasis oli tähtis välja selgitada, kuivõrd on vana versioon vastavuses klientide nõudmiste ja soovidega. Koostati plaan, kuidas võiks välja näha lihtsustatud rakenduse voog, mis on kujutatud joonisel 5. Esmane probleem suuremate klientide silmis oli samuti kehv jõudlus, kuid uute nõuete kogumisel tutvuti ka uue infoväljaga, mida kliendid aruandes näha sooviksid. Sellest tulenevalt pandi tiimina kokku uus nägemus, kuidas lisandunud info aruandesse võiks mahtuda ning kuhu liigutada aruandest eemaldatav, kuid siiski vajalik info. Selgus, et pealkirja all kuvatav tipptunni info, nagu on kujutatud lisas 2, ei peaks enam aruandesse kuuluma, vaid võiks eksisteerida täpsustatud kujul teises vahelehes. Selle asemel peaks tabelisse lisanduma hinnaala tulp, mille abil klient saaks vaadata, milline tipptund kehtib millise mõõtmispunkti kohta, kujutatud lisas 1. Tänu sellele sai andmebaasist eemaldada tipptundide kohta käiva tabeli ning jääkaruande tabelisse tuli lisada tulp hinnaalade kohta. Lõplik

andmebaasi struktuur on kujutatud joonisel 6. Tipptundide jaoks mõeldud uus vaheleht on samuti lõputöö autori poolt loodud ning see on kujutatud lisas 3. Näidatud on ka jääkaruanne vanal kujul lisas 2 ning uuel kujul lisas 1.



Joonis 6. Andmebaasi lõplik struktuur.

Põhiline osa arendusprotsessist toimus Techyoni arenduskeskkonnas, kus andmed on genereeritud küll päriselulisele andmevoo stsenaariumile mõeldes, kuid siiski imiteeritud väärtustega, kuna klientide andmed on konfidentsiaalsed. Konfidentsiaalsusest tingituna on kõik kuvatõmmised teenuse kasutajaliidesest tehtud arenduskeskkonnas. Samuti ei ole seetõttu lubatud kirjeldada täpsemalt teiste mikroteenuste nimesid ega sisu.

Mikroteenuse arendusel oli viimane etapp koodi refaktoreerimine ning testimine erinevate andmetega, mis tähendas teenuse jooksutamist testimisotstarbelises serveris. Teenuse kirjutamise protsessi vältel viidi läbi korduvaid koodianalüüse vanemarendajatega, mis tagasid koodi kvaliteedi ning praktilise osa valiidsuse. Samuti toimusid koosolekud arhitektide ja analüütikutega, millega garanteeriti aruande funktsionaalsuse vastavus kliendi nõuetele. Tuleb arvestada, et suure tarkvarateenuse projekti eluea jooksul toimub pidevaid muutusi tehnoloogilistes valikutes ning lähenemistes, mille käigus vaadatakse kriitiliselt üle ka valminud teenuse koodibaas.

3.3 Teenuse kirjutamise käigus esinenud probleemid

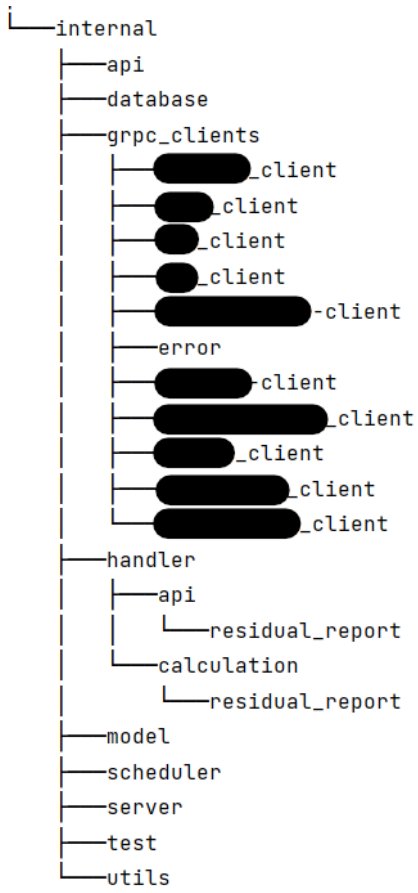
Kuna aruandluse funktsionaalsuse põhimõte põhineb suures osas andmete kokku kogumises paljudest erinevatest mikroteenustest, on praktilise osa raames arendatud mikroteenus sõltuv olemasolevatest andmetest. See aspekt oli ka kõige pidurdavam protsess teenuse arendamise juures. Uuenenud nõuete tõttu tekkis vajadus pärida andmeid mikroteenustest, mida vana implementatsioon ei vajanud. Uute päringute tegemise käigus selgus aga tihti, et vajalikke

andmeid nendest teenustest aruande koostamise jaoks kätte ei saadud ning põhjusi selleks oli mitmeid. Näiteks ei oldud õigel kujul andmeid mõnes teenuses veel genereeritud või sisestatud, kuna varem polnud nende järele vajadust. Mõnel juhul polnud veel mõnda vajalikku API-t täielikult implementeeritud. Lõppfaasis, kui praktilise osa raames arendatud teenus oli käima pandud testimisotstarbelises serveris, selgus, et mõnd vajalikku *back-end* teenust ei jookse samas serveris ning sellisel juhul pidi need esmalt käivitama, andmed genereerima ning alles siis sai uus vahevarakihis asuv teenus komplekteerida korrektsel kujul aruande. Õnneks oli Techyoni tiim väga professionaalne ning kiire nende murede lahendamisel.

Üks probleem oli ka vanemarendajate ning arhitektide ajaline ressurss. Lõputöö raames arendatud mikroteenust kirjutas töö autor ainuisikuliselt ning kiirematel perioodidel, kui arhitektid ja vanemarendajad olid hõivatud kriitiliste probleemide lahendamisega, toimus koodianalüüsi harvemini või mitte nii põhjalikult, mis pärssis teenuse arendamise kiirust ning jooksvat kvaliteeti.

3.4 Teenuse struktuur

Teenus on üles ehitatud lähtudes autori projekti tiimi tavadest. Peamiselt hõlmab see kaustade jaotust, kausta ja failide nimetamissüsteemi ning teenuse korrektse käivitamisega seotud faile (“Dockerfile”, “docker-compose.yaml”, “Makefile”), mille jaoks on ühtne mall. Juurkausta kuulub kõik versioonihalduse (*Git*) ning rakenduse käivitamisega seonduv. Tasub märkida, et teenuses on eraldi failid “.env” ning “app.env”. Fail nimega “app.env” hoiustab endas lokaalse jooksutamise jaoks vajalikke keskkonnamuutujaid ning “.env” on mõeldud automatiseeritud tarnimise jaoks pilves. “README.md” faili on kaasatud ka juhend sisemiseks kasutuseks. Selles on kirjeldatud põgusalt teisi vajalikke mikroteenuseid ning ka jääkaruande sisu.

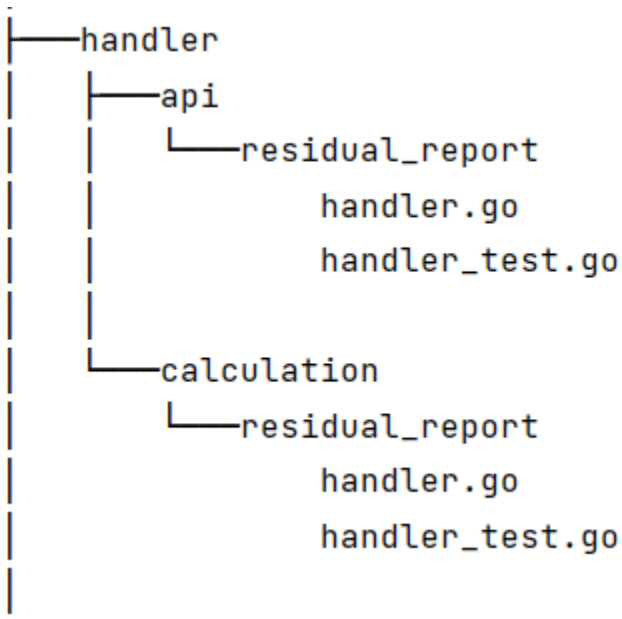


Joonis 7. Kausta “internal” sisu kuvatud puuna (teiste mikroteenuste nimed peidetud).

Juurkaustas asuvasse “database” kausta kuuluvad SQL skriptid, mis on seotud andmebaasi loomisega. Sellele järgnevas “integrationtest” kaustas asuvad andmebaasi testid. Siinkohal peab tõdema, et arendatud teenuse kaetavus testidega on madal, mis on võimalus jätkuarendusteks. Kogu aruande koostamise loogika on paigutatud “internal” kausta. Töö autor lisab, et selline kausta struktuur on osaliselt teenuse kirjutamise lõpuks juba tiimi kooditavadest välja juuritud, selle asemel paigutatakse edaspidi “internal” kausta sees olevad kaustad otse juurkausta. Kausta “internal” sisu on kujutatud joonisel 7.

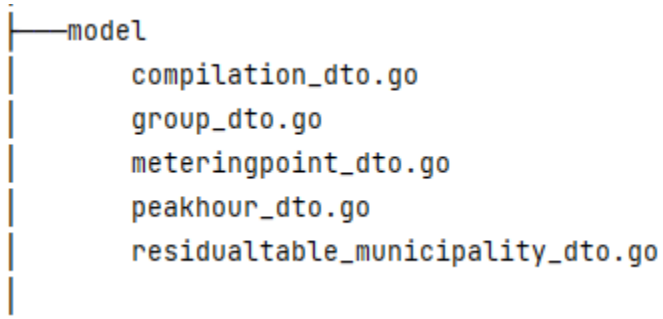
*K*austa “internal” sees asuv “api” kaust hoiab endas “api.go” faili, milles asubki teenuse lõpp-punkt (ingl k *endpoint*) “GetResidualReport”, mis käivitab protsessi, et parameetrite järgi aruande andmed kokku koguda ning tagastada. Kausta “internal” kuulub veel üks kaust nimega “database”, mille sees asub andmebaasi lähtestamine ning ka funktsioonid, mis andmebaasist andmeid pärivad. Kaustas “grpc_clients” asub vajalik loogika, et luua ühendused teiste vajaminevate teenustega. Iga teenuse faili sees defineeritakse ka meetodid API päringu koostamiseks. Peamine loogika andmetöötluse ning API vastuse koostamise jaoks asub

“handler” kaustas, mis on kujutatud joonisel 8. Selle sisse on paigutatud vastavalt “calculation” ning “api” alamkaustad, mille sees olevad “handler.go” failid sisaldavad endas mahukaid “Handle” funktsioone. “Calculation” puu sees asuva arvutusprotsessi lõpuks sisestatakse andmed tabelisse ning “api” puu sees oleva “Handle” meetodi tulemusena päritakse andmeid tabelist.



Joonis 8. Kausta “handler” sisu.

Nii arvutusprotsessi kui API päringu vastuse moodustamise käigus osutus vajalikuks luua mitmetasemelisi (ingl k *nested*) andmetüüpe ning nende implementeerimiseks on loodud joonisel 9 kujutatud “model” kausta Golangile omased *struct* tüüpi muutujad, mida kasutatakse kui *DTO*-sid (*data transfer object*), millel saab välju väärtustada ning nende sisu pärida (tuntud ka kui *get* ja *set* meetodid).



Joonis 9. Kausta “model” sisu.

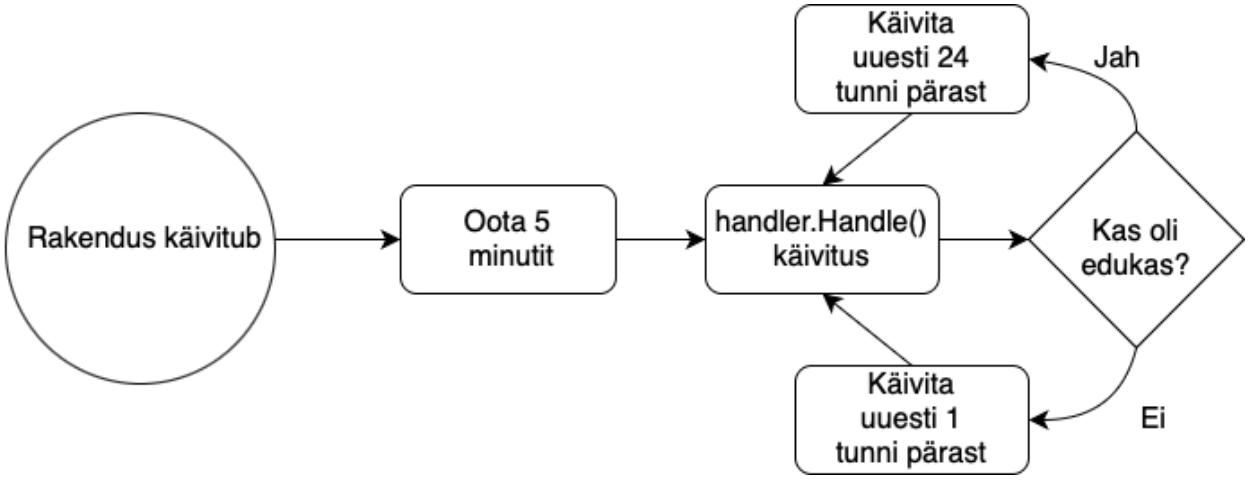
Golangi teenust käivitatakse “main.go” failist jooksutades “main” funktsiooni. Selle funktsiooni sees luuakse ühendused teiste mikroteenustega, käivitatakse andmebaas ning teenuse enda server. Erilist tähelepanu tasub pöörata joonisel 10 oleva meetodi kehas asuvale eelviimasele reale, mis käivitab tööde plaanuri (ingl k *scheduler*).

```
func main() {
    starter.Init( name: "[redacted]", GitCommit, team: "[redacted]")

    grpc_clients.Init()
    aLocker := locker.NewLocker()
    dbClient := database.Init(aLocker)
    handler := residual_report.NewResidualReportHandler(dbClient)
    scheduler.Init(handler)
    server.StartGrpcServer(dbClient)
}
```

Joonis 10. Teenuse “main” funktsioon.

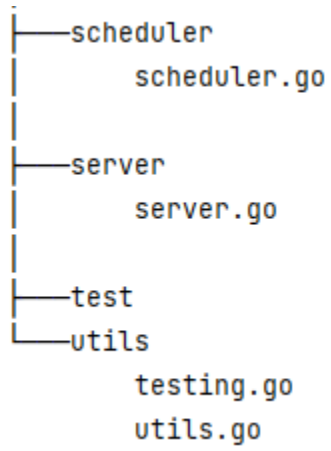
Selle “Init” meetodi sees alustatakse andmetöötluse protsessi, kuid väikese viivitusega. Kuna arendatud teenus loob ühendusi paljude teiste mikroteenustega ning pärib neilt andmeid, on tähtis, et kõik vajalikud teenused töötaksid ning oleks valmis päringutele vastama. Seetõttu ootab teenus käivitumisel viis minutit, enne kui jooksutab andmetöötlusega tegelevat “Handle” meetodit. Tähtis on see seetõttu, et Techyoni keskkondades on iga päev kõikide teenuste planeeritud kellaajaline taaskäivitamine ning sel viisil saab veenduda, et ühendused luuakse.



Joonis 11. Plaanuri loogika.

Eeldusel, et esimene “Handle” funktsiooni käitamine oli edukas, saab edaspidi seda funktsiooni käitada iga kahekümne nelja tunni tagant, et andmeid uuendada. Kui protsess ei lõppenud

edukalt, proovitakse tund aega hiljem uuesti. Kuna Techyoni tarkvara jooksutatakse pilves, tuleb arvestada ka teenust jooksutava kapsli (ingl k *pod*) elueaga, mistõttu on teenuses implementeeritud süsteem, mis on suuteline tuvastama eelmist "Handle" meetodi käitamisaega ka juhul, kui need on erinevate kapslite sees toimunud. Plaanuri lihtsustatud loogika on kujutatud joonisel 11.



Joonis 12. "Scheduler", "server", "test" ja "utils".

Joonisel 12 kujutatud kaustas "server" luuakse teenuse enda server ning ühendused teiste vajalike mikroteenustega. Kaust "utils" hoiab endas abifunktsioone, mis leiavad teenuses korduvat kasutust, ning testimiskeskkonna jooksutamise koodi, mis hetkel on tühi.

4. Valminud lahendus

Arenduse tulemusena on valminud uuel taustaloogikal põhinev ning uuetele nõuetele vastav aruande implementatsioon. Järgnevalt vaadatakse arendusprotsessi tulemusi ning käsitletakse projekti liikmetelt ja klientidelt tulnud tagasisidet. Lisades on näidatud nii vana lahenduse kasutajaliidese vaade kui ka uue lahenduse vaade.

4.1 Tulemused

BloomRPC tööriista kasutades võrreldi vana ning uue API ajalist sooritust. Tulemuste erinevus siinkohal oli märkimisväärne, sest uuelt API-lt saadi vastus ~250 ms jooksul, samas kui vana API tagastas eduka vastuse alles pärast 8 sekundit. See on rohkem kui 30-kordne vahe ning kui lähtuda ainult API ajalisest sooritusest, saab töö tulemust lugeda väga edukaks. Kui lähtuda kliendi perspektiivist, tuleb soorituse mõõtmiseks liita API vastusele kulunud ajale lisaks kasutajaliidese laadimise aeg, mis lisab ca 500 ms protsessi kogupikkusele - loodetavasti paraneb ka see jätkuarenduste käigus.

Võrdleme tulemusi kasutajaliidese seisukohast. Vana implementatsiooni päringu vastuse saamine (*getResidualReport*) võttis veebirakenduses aega umbes 10 sekundit. Töö käigus arendatud teenuse päringu (*GetResidualReport*) tulemusena võtab nüüd tabeli kasutajaliideses laadimine aega ca 800 ms. See tähendab, et andmete laadimine kasutajaliideses toimus üle kümne korra kiiremini.

4.2 Uue teenuse integreerimine veebirakendusse

Arendatud teenuse tarbekeskkonda integreerimine toimus kahes osas. Esmalt käivitati värskelt arendatud teenus tarbekeskkonnas ning loodi vana implementatsiooni kasutajaliidesest koopia, mis kohandati kasutama uut mikroteenust. Uus versioon oli leitav ainult neile, kes teadsid õiget veebiaadressi, st seda ei saanud navigatsiooni menüüst valida nii nagu vana versiooni. Sel viisil sai tagada, et veebilehele pääsesid ligi ainult valitud inimesed ning seda perioodi kasutati tarbekeskkonnas andmete kontrollimiseks ning valideerimiseks. Sel perioodil viidi läbi ka muudatusi kasutajaliideses, mis kajastuvad aruande uue implementatsiooni piltidel lisas 1.

Olles veendunud kõikide muudatuste kasulikkuses ning andmete õigsuses, sai läbi viia täieliku integreerimise ning vana versiooni eemaldamise kasutusest. Selles faasis asendati

navigatsiooni menüüs vana implementatsioon uuega ning eemaldati koodibaasist vana lahenduse kood. Ka kliente teavitati sellest uuendusest.

4.3 Tagasiside

Kuna aruandluse funktsioon on osa *nanoFlow* tootest, mida kasutatakse hetkel veel üksikute Techyoni klientide poolt, on kirjutamise hetkel veel tagasiside hulk klientidelt väike. On teada, et ollakse rahul aruande laadimise kiirusega ning ka kosmeetilised muutused võeti soojalt vastu.

Suurem hulk tagasisidet tuli Techyoni müügi- ning analüüsitiimilt, kellelt tuli positiivne tagasiside aruande kohta. Rõhutati taolise uuenduse kasulikkust müügi aspektist. Võrreldes vana versiooniga, mis oli aeglane ja ebastabiilne, on värskelt valminud aruande uus versioon esinduslikum, töökindlam ning näitab suurt hulka andmeid silmapilkselt, mis muudab aruande kliendile väärtuslikumaks tööriistaks.

Aprillis 2023. aastal toimus ka ulatuslik konverents, kus esitati muuhulgas ka värskelt arendatud jääkaruannet olemasolevatele ning potentsiaalsetele klientidele. Töö kirjutamise hetkel pole veel teada, kas on lisandunud kliente, kes *nanoFlow* toodet on ostnud, kuid on teada, et on suurenenud huvi aruande vastu.

5. Edasiarenduse võimalused

Järgnevalt vaadatakse kriitilise pilguga värskelt arendatud tarkvara peale ning arutletakse, mida võiks uue mikroteenuse arendamisel teha paremini. Kaalutakse mitmeid aspekte nagu päringute optimiseerimine, teenuste vahelise suhtluse edendamine, testide lisamine. Samuti pakutakse välja optimistlik plaan, kuidas Techyon võiks olla kasulik ka Eesti riigile.

5.1 Uue mikroteenuse jätkuarendused

Lõputöö kirjutamise hetkel asub aruande funktsionaalsuse kasutajaliidese koodibaas veel Techyoni monoliitses FE koodihoidlas. Juba toimub arutelu selle üle, et ka FE implementatsiooni võiks liigutada loodavasse mikro-*front-end* koodihoidlasse, mis hõlmaks endas kõikvõimalikke aruandeid Techyoni platformi siseselt.

Praegune versioon arendatud mikroteenusest uuendab aruande andmeid kahekümne nelja tunnise intervalli tagant. Potentsiaalne edasiarendus sellele oleks intervalli asendamine Pub/Sub tehnoloogia kasutamisega, uuendades läbi vajalike tellimuste (ingl. k *subscription*) aruandes olevaid andmeid. Töö kirjutamise hetkel on aga välja uurimata, kuidas see teenuse sooritusvõimet või kulu parandaks.

Veel üheks edasiarenduse võimaluseks on suurem testidega kaetavus. Hetkel katavad testid enamikku töötlusprotsessist ning andmebaasi tehingutest, kuid paljud äärejuhtumid ning potentsiaalsed veateated on jäetud testimata. Samuti on FE implementatsioonis väga vähe teste, mis on kirjutamise hetkel projektis läbiv probleem.

Hetkel asub mikroteenuses ainsa funktsionaalsusena iga-aastase aruande koostamine. On veel määramata, kas valitud mikroteenusesse lisatakse veel mõni aruandlusega seotud funktsionaalsus. On välja pakutud võimalus, et uute aruannete lisandumise korral lisatakse nende andmetöötluse implementatsioonid töö raames arendatud mikroteenusesse.

Mikroteenuste arendustsükkel näeb ette pidevat kaasajastamist ja refaktoorimist ning seda ei saa mainimata jätta ka antud teenuse puhul. Nagu mainiti ka struktuuri kirjelduse käigus, saaks teha teistsuguseid valikuid näiteks kaustade ülesehituses. Samuti esineb koodis korduvaid püsikodeeritud sõnesid, mida saaks paigutada eraldi konstantide faili. Parandada saaks ka muutujate nimetamist, milleni loodetavasti jõutakse järgnevates iteratsioonides.

5.2 Techyon tulevikus ning Eesti kontekstis

Töö autoril puudub küll nooremarendajana põhjalik ülevaade kogu Techyoni võimekusest ning töökavast, kuid pikemas perspektiivis on planeeritud arendatavat digilahendust laiendada ka muude Norra kommunaalteenuste nagu vesi, prügivedu, gaas, konteksti. See pakub edasiarenduse võimalusi ka paljudele olemasolevatele ja uutele teenustele.

Hetkel kogub Techyoni platformil kasulikke domeeniteadmisi selles valdkonnas ~25 CGI Eesti arendajat. Tulevikus võib sellistest kogemustest väga palju kasu olla, kui tekib vajadus/võimalus Eesti elektrisüsteemile tervikliku digilahenduse väljaarendamise järele.

Kokkuvõte

Lõputöö teoreetilises pooles tutvustati mikroteenuseid, nendel põhinevat arhitektuuri ning sellega kaasnevaid tehnoloogiaid. Selgitati lähemalt tarkvara ehitamise struktuuri, kus kasutajaliidese ning serveripoolsete teenuste suhtlus toimub läbi rakendusprogrammiliidese vahendaja ehk vahevarakihi. Samuti tutvuti põgusalt Techyoni teenustega Norra elektrisüsteemis. Suurema osa antud tööst moodustas praktiline pool ehk mikroteenuse arendus, mis demonstreeris teoreetilise sisu praktilisust ning rakendust Techyoni kontekstis.

Lõputöö praktilise osa raames arendati mikroteenus programmeerimiskeeles Go, mis vastab nõuetelt Techyoni veebirakenduse iga-aastase aruandluse funktsionaalsusele ehk jääkaruandele. Kirjeldati vana lahenduse puudujääke ning selgitati, kuidas uus lahendus puudujäägid lahendab, samas pakuti välja ka potentsiaalsed edasiarenduse võimalused uuele lahendusele.

Viidatud kirjandus

1. AKIT - Andmekaitse ja infoturbe leksikon. https://akit.cyber.ee/ (vaadatud 18.04.2023)
2. Andrawos, M., & Helmich, M. (2017). *Cloud Native Programming with Golang: Develop microservice-based high performance web apps for the cloud with Go*. Packt Publishing Ltd.
3. BairesDev. (2022). Why Golang is so Fast: Performance Analysis. BairesDev. https://www.bairesdev.com/blog/why-golang-is-so-fast-performance-analysis/
4. Brown, K. E., & Woolf, B. (2016). Implementation patterns for microservices architectures. PLoP '16 Proceedings of the 23rd Conference on Pattern Languages of Programs, 7.
5. CGI. CGI IS Suite - Utility Solutions. CGI. https://www.cgi.com/no/nb/loesninger/is-suite (vaadatud 18.04.2023)
6. Google Cloud. Go on Google Cloud. Google Cloud. https://cloud.google.com/go (vaadatud 18.04.2023)
7. Kamal Singh. (2021). Microservices Core Principles. Medium. https://medium.com/@chkamalsingh/microservices-core-principles-80e427261bc8
8. Kofi Group. (2021). 7 reasons why VueJS is so popular. Kofi Group. https://www.kofi-group.com/7-reasons-why-vuejs-is-so-popular/
9. Microservices.io. API Gateway Pattern. Microservices.io. https://microservices.io/patterns/apigateway.html (vaadatud 18.04.2023)
10. Microsoft. Direct client-to-microservice communication versus the API Gateway pattern. Learn | Microsoft Docs. https://learn.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern (vaadatud 18.04.2023)
11. Mõõtepunkt | Energiatalgud. https://energiatalgud.ee/moisted/mootepunkt (vaadatud 18.04.2023)
12. Netflix Technology Blog. (2022). Optimizing the Netflix API. Medium. https://netflixtechblog.com/optimizing-the-netflix-api-5c9ac715cf19
13. OpenLegacy. (2021). Monolithic Application. OpenLegacy Blog. https://www.openlegacy.com/blog/monolithic-application

14. Sathya's Blog. (2021). Micro Frontends. WordPress. https://sathyalog.wordpress.com/2021/01/03/micro-frontends/
15. Semaphore. (2017). Monolith vs. Microservices Architecture. SemaphoreCI Blog. https://semaphoreci.com/blog/monolith-microservices
16. Statnett. For stakeholders in the power industry. Statnett. https://www.statnett.no/en/for-stakeholders-in-the-power-industry/ (vaadatud 18.04.2023)
17. The Go Programming Language. Go. https://go.dev/ (vaadatud 18.04.2023)

Lisad

RESIDUAL REPORT FOR 2023

From 2012 To 2021

MUNICIPALITY	PRODUCTION MP ID	ALTERNATIVE ID	GRID AREA	PRICE AREA	PT [KW]	PRICE NOK	RESULT NOK	AVERAGE	2021	2020	2019	2018
0819 NOME	707057500081876700		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	-
	707057500081779490	TEST76453	MTEL1		10,00	1,10000	48 224,00	43 840	43 800	43 920	43 800	3 240
	707057500081879893		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	-
	707057500081870227		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	-
	707057500081876465		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	-
	707057500081877264		MTEL1		0,00	1,10000	-	-	-	-	-	-
	707057500081784920		MTEL1		1 000,00	1,10000	48 180,00	43 800	43 800	2 400	-	-
	707057500081877066		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	-
	707057500081879824		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	-
SUM:					1 010,00	1,10000	96 404,00	87 640	87 600	46 320	43 800	3 240
0821 BØ	707057500081875420		MTEL1		0,00	1,10000	-	-	-	-	-	-
	707057500081778134		MTEL1		0,00	1,10000	48 221,90	43 838	43 800	43 920	43 794	-
	707057500081778684		MTEL1	NO2	6,46	1,10000	-	-	8 880	-	-	-
SUM:					6,46	1,10000	48 221,90	43 838	52 680	43 920	43 794	0
0822 SAUHERAD	707057500081778677		MTEL1		0,00	1,10000	48 180,00	43 800	43 800	-	-	-
	707057500081872146		MTEL1		0,00	1,10000	-	-	9 477	-	-	-
	707057500081778875		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	3 600
SUM:					0,00	1,10000	48 180,00	43 800	53 277	0	0	3 600
1106 HAUGESUND	707057500081878551		MTEL1	NO2	0,00	1,10000	-	-	-	-	-	-
SUM:					0,00	1,10000	-	-	0	0	0	0
TOTAL SUM:					1 016,46	1,10000	192 805,91	175 278	193 557	90 240	87 594	6 840

Lisa 1. Jääkaruande kasutajaliidese uus kuju.

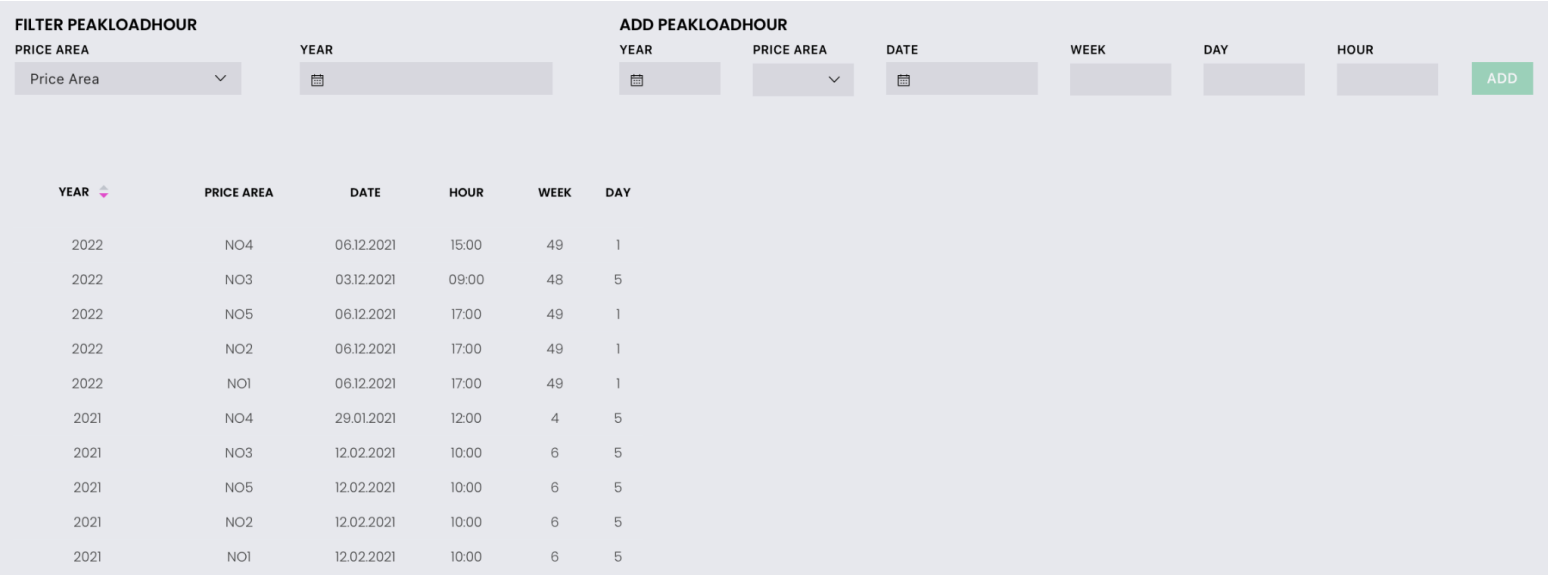
From: 2011 To: 2022

Residual report for 2022

Peakload hour: 12.02.2021 10:00:00

MUNICIPALITY	PRODUCTION MP ID	ALT ID	GRID AREA	PT [KW]	PRICE NOK	RESULT	AVERAGE	2020	2019	2018	2017
0821 Bø	707057500081872856	707057500081872856.4	MTEL1	10.00	360.12344	-	-	-	-	-	-
	707057500081872146		MTEL1	0.00	360.12344	-	-	-	-	-	-
	707057500081778677		MTEL1	-	360.12344	-	-	-	-	-	3720.00
	707057500081870227		MTEL1	-	360.12344	-	-	-	-	-	-
	707057500081778875		MTEL1	-	360.12344	-	-	-	-	3600.00	-
SUM:				10.00	360.12344	-	-	-	-	3600.00	3720.00
0821 Bø	707057500081784920		MTEL1	1000.00	360.12344	-	-	2400.00	-	-	-
SUM:				1000.00	360.12344	-	-	2400.00	-	-	-
0822 Sauherad	707057500081778134		MTEL1	6.46	360.12344	15793984.28	43857.14	43920.00	43794.28	-	1680.00
	707057500081781882		MTEL1	2.00	360.12344	-	-	-	-	-	-
	707057500081779490	test76453	MTEL1	10.00	360.12344	15795014.24	43860.00	43920.00	43800.00	3240.00	-
SUM:				18.46	360.12344	31588998.52	87717.14	87840.00	87594.28	3240.00	1680.00
TOTAL SUM:				1028.46	360.12344	31588998.52	87717.14	90240.00	87594.28	6840.00	5400.00

Lisa 2. Jääkaruande kasutajaliidese vana kuju.



Lisa 3. Tipptunde kajastav vaheleht.

Litsents

Mina, **Robert Leht**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose "**Mikroteenuse loomine projekti Techyon vahevarakihti**", mille juhendaja on Helle Hein, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Robert Leht
07.05.2023