TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

William Straus 185692IADB

# REST-põhiste mikroteenuste väljatöötamine Amazoni pilveplatvormil krediidiriski hindamiseks

Bakalaureusetöö

Juhendaja: Einar Kivisalu
MSc

Tallinn 2021

# Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: William Straus

14.05.2021

# Annotatsioon

Käesolev bakalaureusesetöö käsitleb REST-standardil põhineva mikroteenuste API loomist, mis võimaldab välistest registritest küsida ettevõtte äriprotsessi jaoks vajalikke andmeid, neid töödelda ning kasutajale JSON-formaadis tagastada.

Olulisemad käsitletavad probleemid on päringute tegemine välistesse registritesse SOAP-i ja X-tee kaudu, saadud vastuste teisendamine JSON-formaati, vastuste salvestamine andmebaasi ning JSON-vastuste tagastamine kasutajale.

Töö tulemusena valmis toimiv API, mida ettevõte kasutab klientide krediidivõimekuse hindamiseks ning äriliste otsuste tegemiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 5 peatükki, 15 joonist, 1 tabelit.

# Abstract

Development of REST-Based Microservices on the Amazon Cloud Platform for Credit Risk Assessment

The goal of the bachelor's thesis is to create a REST-based API of microservices which requests data from outside registers, processes it and returns it in JSON format to the user. The API is created for the start-up company Hoovi which offers home rent prepayment and rent guarantee for landlords, as well as business loans for companies. Because the company cannot offer credit to insolvent clients, it needs to check their background and make an informed decision about whether to sign a contract with them.

The main problems are the making of requests to external registers via SOAP and X-road, the conversion of the data into JSON format, the caching of the data into a database and the returning of the JSON-responses to the user. Six registers were integrated into the API: Property Register, Building Register, Creditinfo, taust.ee, Pension Center and Population Register.

As a result of the work, a fully functional API was completed, which the company uses for making assessments about the credit risk of potential customers and for making business decisions.

The thesis is in Estonian and contains 47 pages of text, 5 chapters, 15 figures, 1 table.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| API | *Application Programming Interface,* rakenduse programmeerimise liides |
| *API Gateway* | AWS-i teenus API-de loomiseks, kasutusele võtmiseks, hooldamiseks, jälgimiseks ja turvamiseks [1] |
| AWS | Amazon Web Services |
| *Back-end* | Infosüsteemi osad, milles hoitakse ja töödeldakse andmeid [2] |
| *Backlog* | Tähtsuse järjekorda seatud nimekiri ülesannetest, mida kollektiiv plaanib järgmisena lahendada [3] |
| *CloudWatch* | AWS-i kuuluv jälgimisteenus [4] |
| *DynamoDB* | AWS-i kuuluv NoSQL (mitte-SQL) andmebaasiteenus [5] |
| *Endpoint* | Lõpp-punkt; sidekanali lõpp [6] |
| *Front-end* | Infosüsteemi osad, millega kasutaja otse suhtleb [2] |
| Git | Versioonikontrolli süsteem [7] |
| IDE | *Integrated Development Environment*, integreeritud arenduskeskkond |
| JSON | *JavaScript Object Notation,* tekstipõhine andmete vahetamise formaat [8] |
| *Lambda* | AWS-i kuuluv serverivaba arvutusteenus [9] |
| *Query parameter* | URL-i lõppu lisatud parameeter [10] |
| REST | *Representational State Transfer,* arvutisüsteemide vahelise suhtluse arhitektuuriline stiil [11] |
| SOA | *Service-Oriented Architecture,* teenustele orienteeritud arhitektuur; arhitektuurne lähenemine, mille järgi rakendused kasutavad võrgus kättesaadavaid teenuseid [12] |
| SOAP | *Simple Object Access Protocol,* XML-i põhine sõnumiedastusprotokoll [13] |
| URL | *Universal Resource Locator,* universaalne ressursilokaator |
| WSDL | *Web Services Description Language,* XML-i formaat võrguteenuste kirjeldamiseks [14] |
| XML | *Extensible Markup Language, „*meta-keel“, mille abil saab luua märgendikeeli [15] |

| | |
|---|---|
| X-tee | Keskkond, mis võimaldab turvalist ja tõestusväärtust tagavat internetipõhist andmevahetust riigiasutuste vahel ja erasektoriga [16] |

# Sisukord

# Jooniste loetelu

# Tabelite loetelu

# 1 Sissejuhatus

Käesolev bakalaureusetöö on kirjutatud tehnilise lahenduse põhjal, mis on mõeldud registritest krediidivõime hindamiseks vajalike andmete hankimiseks. Selle lõi autor praktika käigus iduettevõttes Hoovi (Hoovi Group OÜ, siin ja edaspidi Hoovi) äriprotsessi hõlbustamiseks.

## 1.1 Probleem

Andmed on tänapäeva firmade äriprotsessis väga olulised. Neid saadakse paljudest välistest allikatest. Paraku on andmed tihti erineva formaadiga ning nende kasutamine seetõttu tülikas. Registritega suhtlemiseks kasutatakse protokolle nagu REST ja SOAP, samuti hoopis komplekssemaid lahendusi nagu X-tee. Lisaks välistele registritele kasutatakse ka ettevõttesiseseid andmebaase. Kuna tihti antakse registritest saadud kirjed rakendustele sisendiks, siis võib formaatide rohkus suuri raskusi valmistada.

Just selline probleem tekkis autoril praktika käigus ettevõttes Hoovi. Hoovi pakub korteriomanikele üüri ettemaksu ja üürigarantiid, samuti ettevõtetele ärilaenu. Nende teenustega kaasneb oluline majanduslik risk: kui ettevõte pakuks igale huvitatud kinnisvaraomanikule üüri ettemaksu või üürigarantiid, siis võiks Hoovi saada suurt majanduslikku kahju. Firma ei taha pakkuda üüri ettemaksu korteriomanikule, kelle üürnik on maksejõuetu või peaaegu maksejõuetu. Sel juhul saab omanik küll oma raha kätte, kuid Hoovil tekib kahju, kuna firma ei suuda üürnikult üüri välja nõuda. Samuti ei tasu Hoovil anda ärilaenu ettevõttele, kes seda tagasi maksta ei suuda. Seega on krediidi taotlejatele vaja teha taustakontroll, et veenduda nende maksevõimes.

Taustakontrolli tegemiseks on vaja kasutada väliseid registreid. Et aga mitu vajalikku registrit kasutab kas SOAP-protokolli või X-teed, siis ei saanud Hoovi neid otse kasutusele võtta, vaid tuli luua päringuid vahendav ja töötlev rakendus. Selle probleemi lahendamist käsitlebki käesolev bakalaureusetöö.

## 1.2 Eesmärk

Käesoleva töö eesmärgiks on luua REST põhimõtteid järgiv API, milles sisalduvate mikroteenuste abil oleks välistesse registritesse lihtne päringuid teha ja sealt vastuseid saada. Seejuures peaks vastuste standardne formaat tagama selle, et neid annab automaatselt edasi töödelda. Need andmed võimaldavad ettevõttel teha kaalutletud otsuse, kas kinnisvaraomanikule tasub ettemaksu teha või ettevõttele ärilaenu anda. REST-standardi kasutamise oli autorile ette kirjutanud ettevõte, samuti ka välised andmeallikad, mida integreerida.

## 1.3 Ülevaade tööst

Töö käigus kirjutati kood järgnevate registrite integreerimiseks:

- Kinnistusraamat
- Julianus Inkasso
- Rahvastikuregister

Järgnevale registrit ei integreeritud otse, vaid ettevõte küsis andmed tervikfailina välja ja salvestas need enda andmebaasi, kust sobivate kirjete saamiseks tuli mikroteenus kirjutada:

- Ehitisregister

Järgnevaid teenuseid arendati edasi: andmete töötlemise osa kirjutasid ettevõtte teised töötajad, autor lisas neile REST-*endpointid*:

- Pensionikeskus
- Creditinfo

# 2 Metoodika

Käesolevad peatükis antakse ülevaade töös kasutatud registritest, tööriistadest ning tööprotsessist.

## 2.1 Mikroteenused

Käesolevas töös kirjeldatav API loodi üldjoontes mikroteenuste arhitektuuri järgides. Portaali microservices.io sõnul on seda arhitektuuristiili järgivad süsteemid [17]:

- Kergesti hooldatavad ja testitavad
- Nõrgalt omavahel seotud komponentidega
- Iseseisvalt kasutusele võetavad
- Korraldatud ärivajadustele vastavalt
- Väikese meeskonna vastuse all

Larrucea, Santamaria, Colomo-Palaciose ja Eberti sõnul tekkis selline mudel 2000-ndate lõpus teenustele orienteeritud arhitektuuri (*Service-Oriented Architecture*, lühendina SOA) põhjal [18]. Nende sõnul seisneb see „keeruliste rakenduste eraldamises väikesteks tükkideks“, mis „parandab jõudlust“ [18].
Hoovi jaoks arendatud API koosneb kuuest mikroteenusest, mis neid põhimõtteid järgivad.

## 2.2 X-tee

X-tee on Riigi Infosüsteemi Ameti (RIA) sõnul „keskkond, mis võimaldab turvalist ja tõestusväärtust tagavat internetipõhist andmevahetust riigiasutuste vahel ja erasektoriga“ [16]. Turvalisus tähendab, et tagatud on konfidentsiaalsus, käideldavus ja terviklus [19]. X-teed peab üleval RIA (Riigi Infosüsteemi Amet), kes Anto Veldre sõnul „võtab vastu uusi liikmeid“, seob sertifikaatidega kindlad õigused ning jälgib X-tee kasutust [20]. Veldre sõnul loodi X-tee selleks, et vältida suure keskse riikliku andmebaasi loomist, millega kaasneks suur turvarisk (kuna andmelekke korral saadakse inimese kohta korraga palju andmeid kätte) [21]. Kui aga selline suur andmebaas oleks dubleeritud paljudes asutustes, siis tõenäoliselt läheksid andmed inimese kohta eri asutustes aja jooksul sünkroonist välja [21]. X-tee loodi eeldusel, et ideaaljuhul hoiab iga asutus inimese kohta

vaid tollele asutusele vajalikke andmeid, teised asutused peaksid aga vajadusel saama neid andmeid turvaliselt välja küsida [21].

Riigiasutus või eraettevõtte teeb liitub X-teega ja saab andmeteenuse osutajaks, tehes oma andmebaasi kättesaadavaks [20]. Kui eraisik või ettevõte soovib X-tee kaudu andmeid küsida, esitab ta RIA-le X-teega liitumise taotluse ning sõlmib huvipakkuva registri pidajaga kasutuskokkuleppe [20].

Nii andmeteenuse osutajal kui ka passiivsel kasutajal (kes küsib, aga ei paku andmeid) ehk kliendil on seadistatud turvaserver, mis krüpteerib päringuid ja vastuseid ning suunab need õigele IP-aadressile [20]. Igas turvaserveris on logi, kus peetakse arvestust kõigi päringute üle; logidega ei ole võimalik tagantjärele manipuleerida, ilma et seda avastataks (terviklus on kaitstud) [19].

X-tee ametliku juhendi järgi küsib nii kliendi kui ka andmeteenuse turvaserver „iga minut keskserverilt globaalset konfiguratsiooni, kus on kirjeldatud kõik liikmed, turvaserverid ja alamsüsteemid“ [22]. Samuti pöörduvad turvaserverid usaldusteenuse poole sõnumite ajatembeldamiseks ning sertifikaatide kehtivuskinnituse saamiseks [22].

Antud töö käigus kasutati X-teed Pensionikeskuse ja Rahvastikuregistriga suhtlemiseks (vt Joonis 10 ja Joonis 11).

## 2.3 Registrid

Töö käigus integreeriti kuus registrit. Neli nendest olid riiklikud: e-Kinnistusraamat, Rahvastikuregister, Pensionikeskus ja Ehitisregister. Kaks registrit olid eraomandis: taust.ee ja Creditinfo.

### 2.3.1 Kinnistusraamat

Registrite ja Infosüsteemide Keskuse (RIK) e-Kinnistusraamat „sisaldab andmeid kõigi Eesti kinnistute kohta“ [23]. Kinnistuteks loetakse Justiitsministeeriumi sõnul kinnisasju (ehk maatükke), hoonestusõigusi, korteriomandeid ning korterihoonestusõigusi [24]. RIKi sõnul saab selles igaüks ise kiiresti ja lihtsalt kontrollida kinnistute üldandmeid, pindala, omanikke, piiranguid, kinnistuid koormavate hüpoteekide andmeid jm [23].

Sellel on oma kasutajaportaal, ent antud juhul kasutati XML-teenust, millega suhtlemine toimub SOAP-protokolli alusel. RIK kirjeldab mitut päringut, mida kasutada saab:

1. Objektide arv: tasuta eelpäring, mis isikukoodi alusel tagastab ainult isikule kuuluvate kinnistute arvu [25].
2. Isiku kinnistud: tasuline päring, mis tagastab isikukoodi alusel detailsema info isikule kuuluvate kinnistute kohta, sealhulgas kinnistu liigi, nime, registriosa numbri, katastritunnuse, pindala ning aadressi [25].
3. Kinnistu detailandmed: tasuline päring, millega registriosa numbri järgi tagastatakse eelmisest päringust veelgi täpsemaid andmeid; andmeid tagastatakse valitud jagude alusel [25]:

    3.1. 1. jagu: piiratud asjaõigused [23]
    3.2. 2. jagu: omaniku info [23]
    3.3. 3. jagu: piiratud asjaõigused, servituudid, kasutusõigused, ostueesõigused, märked [23]

E-kinnistusraamat võimaldab kontrollida, kas ärilaenu taotlejale kuulub kinnisvara, mida saaks tagatisena kasutada.

### 2.3.2 Ehitisregister

Ehitisregistri sõnul on registri eesmärgiks „ehitatavate ja kasutatavate ehitiste kohta teabe koondamine, hoidmine ja avalikustamine“ [26]. Register sisaldab hulgaliselt tehnilisi parameetrid nii hoone kohta tervikuna kui ka osade kohta, nagu näiteks:

- Ehitisalune pind
- Elamispind
- Energiaklass
- Korterite arv
- Tubade arv
- Korruste arv
- Seisund
- Peamine sihtotstarve

Ehitisregistril on nii X-tee teenused kui ka avaandmete API [27], mille kasutamiseks registreerida pole vaja, ent kust üksikkirjeid pärida ei saa. Hoovi kasutas avaandmete API-t. Sealt saadi kõigi ehitiste andmed ühe failina ning salvestati need DynamoDB andmebaasi.

Ehitisregistri abil saab hinnata kinnisvaraobjekti ligikaudset väärtust, mille abil saab otsustada, kas objekt on ärilaenu tagatiseks sobiv. Autorile teadaolevalt pole alternatiivset andmebaasi sellekohase info saamiseks ning riikliku andmebaasi dubleerimisel pole ka mingit praktilist mõtet.

### 2.3.3 Taust.ee

Firma kodulehe andmeil haldab portaali Taust.ee ettevõte nimega Krediidiregister OÜ, mis kuulub Julianus Grupp OÜ-le [28]. Ettevõtte sõnul on portaali eesmärgiks „koguda erinevatelt osapooltelt kokku taustainfo nii era- kui ka juriidiliste isikute kohta, seda töödelda ja huvitatud isikutele vahendada“ [28]. Taust.ee valdab enda sõnul infot „üle 150 000 maksehäire“ kohta [28]. Portaalil on olemas SOAP-liidestus, mis isikukoodi või ettevõtte registrikoodi alusel tagastab järgnevad andmeväljad [29]:

- Nimi
- Nõude alguskuupäev
- Nõude lõpukuupäev
- Nõude suurus
- Nõude staatus
- Tegevusvaldkond

Liidestust on tarvis üürniku või ärilaenu taotleja maksejõulisuse kontrolliks: kui tal on palju võlgu, siis pole ta tõenäoliselt usaldusväärne üürnik/laenaja.
Teenuse peamine alternatiiv on Creditinfo, mida ettevõte samuti kasutab.

### 2.3.4 Creditinfo

Creditinfo on enda sõnul „Eesti suurim eraisikute ja ettevõtete krediidiinfo andmebaas ning Ametlik Maksehäireregister“ [30]. Creditinfost küsitakse infot maksehäirete ja ametlike teadaannete kohta. Viimased on vajalikud üürniku või laenutaotleja usaldusväärsuse kontrollimiseks sarnaselt Taust.ee-ga.

### 2.3.5 Rahvastikuregister

Rahvastikuregister on Siseministeeriumi sõnul „andmekogu, mis koondab Eesti kodanike, Eestis elukoha registreerinud Euroopa Liidu kodanike ja Eestis elamisloa või

elamisõiguse saanud välismaalaste peamisi isikuandmeid“ [31]. Registris on ministeeriumi sõnul teave näiteks isiku ees- ja perekonnanimest, sünniandmetest, isikukoodist, kodakondsusest, elukohast ja perekonnaseisust [31]. Rahvastikuregistri abil kontrollib Hoovi, kas teenust taotlev isik on elus ning kas ta on teovõimeline. Registrile pole alternatiivi, kuna keegi peale riigi ei saa nii privaatseid isikuandmeid koguda ja välja jagada.

### 2.3.6 Pensionikeskus

Pensionikeskus on Nasdaqi sõnul „riigi infosüsteemi kuuluv andmekogu kogumispensionide seaduses sätestatud kohustuslike (pensioni II sammas) ja vabatahtlike (pensioni III sammas) pensionifondide osakute ning nendega tehtavate toimingute registreerimiseks“ [32]. Hoovi küsib Pensionikeskusest infot inimeste pensionisammaste sissemaksete kohta, et arvutada välja isiku sissetulek. Selle abil annab kontrollida, kas üürnik või laenutaotleja on piisavalt maksevõimeline, et talle korterit välja üürida või laenu anda.

## 2.4 Tööriistad

Antud alampeatükis käsitletakse lühidalt peamisi tehnilisi vahendeid, mida arendusprotsessis kasutati.

### 2.4.1 Python

Rakenduse kood kirjutati programmeerimiskeeles Python. Python on üks levinumaid programmeerimiskeeli; 2020. aastal oli ta kahe eri edetabeli järgi populaarsuselt kas esimesel või kolmandal kohal [33]. Keele lõi portaali W3Schools andmeil Guido van Rossum 1991. aastal ning iseloomustab lihtne süntaks [34]. Python on sama veebilehe järgi interpreteeritav keel [34], mis tähendab, et ta käivitamine pole kuigi kiire. Keeles kasutatakse looksulgude asemel tühikuid ning semikooloni asemel reavahetust. Liidestuse loomisel kasutati Pythonit, kuna eelnev osa Hoovi *back-end* süsteemis oli juba Pythonis kirjutatud ning mitme keele kasutamine sama süsteemi piires teeks süsteemi haldamise tarbetult segaseks ja keeruliseks, kuigi AWS-is on võimalik kasutada ka näiteks Javat või C#.

### 2.4.2 Bitbucket

Bitbucket on Atlassiani sõnul „Giti-põhine koodi majutamise ja koostöö tegemise tööriist“ [35]. Kui koodis tehakse muudatusi, siis laetakse uuendatud kood Bitbucketisse, et

1. kood oleks turvaliselt varundatud;
2. teised tiimiliikmed pääseksid selle ligi;
3. saaks vajadusel ennistada varasemat seisu;
4. koodi saaks toru (*pipeline*) kaudu automaatselt käiku lasta.

Bitbucketi võimalikud alternatiivid on portaali g2.com hinnangul GitHub ja Gitlab [36]. Bitbucket on kuni viiele kasutajale tasuta [37]. GitHub oli privaatsete repositooriumite korral varem tasuline, alates 2019. aastast on ka see tasuta [38] [39]. Gitlab on samuti tasuta [40]. Bitbucket on Hoovis algusest peale kasutusel olnud, seetõttu kasutati ka käesoleva töö käigus just Bitbucketit.

### 2.4.3 Amazon Web Services (AWS)

Amazoni sõnul on AWS „lai valik globaalseid pilvepõhiseid tooteid, sealhulgas tööriistad arvutamiseks, majutamiseks, andmete hoidmiseks, analüütikaks, võrkude haldamiseks, mobiilirakenduste loomiseks, arendamiseks“ ja veel muukski [41]. AWS pakub kümneid alamteenuseid, millest töö käigus kasutati ainult mõnda üksikut:

1. Lambda
2. DynamoDB
3. CloudWatch
4. API Gateway

### 2.4.4 Chalice

Chalice on ametliku GitHubi lehe andmeil „raamistik serverivabade rakenduste kirjutamiseks Pythonis“ [42]. Raamistiku abil saab loodud rakendust kergesti Lambda kujul AWS-i paigaldada, võimaldades näiteks REST API luua (vt Joonis 1) [42].

```
from chalice import Chalice

app = Chalice(app_name="helloworld")

@app.route("/")
def index():
       [muu kood]
```

Joonis 1. Kood REST *endpointi* loomiseks Chalice'i abil.

Chalice(app_name="helloworld") viitab, et tegemist hakkab olema Lambda-rakendusega, mille nimi on „helloworld“; *@app.route("/")* on annotatsioon, mis viitab millisel aadressil juurkausta suhtes rakendus asuma hakkab. Viimane on vajalik API Gateways vajaliku *endpointi* loomiseks.

Rakenduse paigaldamiseks AWS-i on vaja esiteks AWS-i seadetes sisestada kasutaja andmed ja õige regioon ning teiseks käsureal sisestada käsk *chalice deploy –stage dev/prod*, kus *dev* viitab arenduse käigus kasutatavale keskkonnale ning *prod* viitab kommertstegevuse käigus kasutatavale keskkonnale.

Chalice teeb arendamise oluliselt mugavamaks, kuna vastasel juhul tuleks õiged komponendid käsitsi AWS-i paigaldada.

## 2.5 Arendusprotsess

Rakenduse arendusprotsess Hoovis oli suhteliselt väheformaalne. Projektihalduse tarkvarana on Hoovis kasutusel Trello. Autor võttis *backlog*ist kõige prioriteetsema tööülesande ning liigutas selle tegemisel olevate ülesannete tulpa (*Doing*). Kood kirjutati PyCharmis, misjärel see laeti üles AWS-i arenduskeskkonda. Rakendust testiti, parandati vead, misjärel see laeti ka teises AWS-i regioonis paiknevasse *live*-keskkonda. Kui rakendus oli ka seal testitud, märkis autor Trellos ülesande tehtuks (liigutati *Development Done* tulpa), misjärel teine meeskonna liige kontrollis ka ülesande teostuse üle. Vajaduse korral suunati kood autorile tagasi parandamiseks. Kui aga parandamist enam vaja polnud, liigutati ülesanne *Live/Done* tulpa. Iga nädal toimusid *online*-koosolekud, kus räägiti tööülesannete edenemisest ning valiti kõige prioriteetsemad tööülesanded järgmiseks nädalaks.

## 2.6 Analoogsed lahendused

Kuna X-tee teenuseid kasutab palju ettevõtteid ja ametiasutusi, siis on sellele kirjutatud kindlasti ka palju integratsioonirakendusi, kuid suur osa nendest pole avalikud. Avalikult kättesaadavad on näiteks järgnevad rakendused:

- Statistikaameti *xGate Client*
  Ameti sõnul on see „Statistikaameti klientidele pakutav lõppkasutajale mõeldud rakendus, mis lihtsustab andmete ja päringute saatmist X-GATE-i x-tee teenustele“ [43]
- TEHIKu (Tervise ja Heaolu Infosüsteemide Keskus) *xroad-gateway*
  Pakub asutuse sõnul ligipääsu Äriregistri ja Töötamise Registri (TÖR) X-tee teenustele [44]
- Riigiportaali *XTR* (X-tee REST)
  Teisendab WSDL-i järgi JSON-sisendi XML-väljundiks, mida saab kasutada X-tee päringu tegemiseks [45]

Kaks esimest rakendust on mõeldud teenustele, mida Hoovi ei kasutanud, viimast aga polnud vaja, kuna õige XML faili loob Pythoni vastav moodul.

# 3 Arendustöö

Loodi kuus mikroteenust, millest annab ülevaate Tabel 1.

Tabel 1. Loodud mikroteenused.

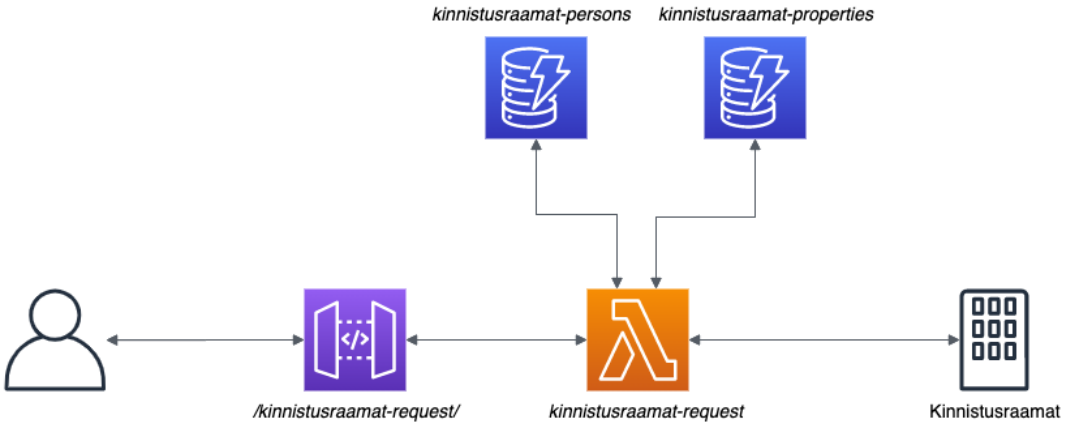
| **Register** | ***Endpoint*** | **Töö käigus loodud** | **Päringute arv** | **Registriga suhtlemise protokoll** |
|---|---|---|---|---|
| Kinnistusraamat | */kinnistusraamat-request/* | Kogu lahendus | 6 | SOAP |
| Ehitisregister | */ehitisregister-request/* | Kogu lahendus | 1 | - (oma andmebaas) |
| Creditinfo | */creditinfo-request/* | Ainult *endpoint* | 1 | SOAP |
| Taust.ee | */julianust-request/* | Kogu lahendus | 1 | SOAP |
| Pensionikeskus | */pensionikeskus-request/* | Ainult *endpoint* | 2 | REST / X-tee |
| Rahvastikuregister | */rahvastikuregister-request/* | Kogu lahendus | 2 | X-tee |

## 3.1 Kinnistusraamat

Ülesande kirjeldus nägi ette, et tuleb luua REST API, mille arendusel kasutatakse eeskujuna juba teise meeskonnaliikme poolt loodud Creditinfo API-t (millele autor lisas *endpointi*). Kinnistusraamatu REST *endpoint* pidi võimaldama kuut päringut:

1. GET: */persons/{ID}/propertyAmount*: isikule kuuluvate objektide arv (eelpäring)
2. GET: */persons/{ID}/properties*: isikule kuuluvate objektide üksikasjalikum teave
3. GET: */properties/{ID}*: jagude 1-4 info kinnisvaraobjekti kohta
4. GET: */properties/{ID}/section1*: jao 1 info kinnisvaraobjekti kohta
5. GET: */properties/{ID}/section2*: jao 2 info kinnisvaraobjekti kohta
6. GET: */properties/{ID}/section3-4*: jagude 3 ja 4 info kinnisvaraobjekti kohta

Kinnistusraamatust tagastatud vastused tuli nii algses XML formaadis kui ka teisendatud JSON-formaadis salvestada AWS DynamoDB andmebaasi, kus oli kaks tabelit: *kinnistusraamat-persons* ja *kinnistusraamat-properties*. Rakenduse üldist struktuuri kirjeldab Joonis 2. Tabelite võtmed pidid moodustuma ID-st ja salvestamise ajast. Salvestamise mõte oli vastuste taaskasutamine teatud aja jooksul kulude vähendamiseks.




Joonis 2. Kinnistusraamat.

*Endpoint*i loomise esimeseks faasiks oli simuleeritud lahenduse loomine. Rakendus pidi tagastama realistliku vastuse, olemata tegelikult veel Kinnistusraamatuga ühendatud. See oli vajalik, testimaks *endpoint*i koostööd süsteemi teiste osadega ning vältimaks arenduse ajal kuutasu maksmist. Teiseks faasiks oli simuleerimise asendamine päris teenuse integratsiooniga.

Esiteks loodi struktuur, mis igale AWS Lambda funktsioonile ette nähtud on, sealhulgas Hoovis loodud tarvikud (failide lugemiseks ja muudeks elementaarseteks toiminguteks). Iga eeltoodud päringu jaoks loodi oma meetod, seega 6 meetodit, mille sees küsiti, kas andmebaasis on hiljuti lisatud vastuseid, ning kui neid polnud, tagastati simuleeritud vastus (hiljem juba tehti päring Kinnistusraamatusse). Kirjutati ka abimeetodid andmebaasi kirjutamiseks, sealt kirje küsimiseks ning simuleeritud vastuse tagastamiseks. Eraldi failidesse kirjutati simuleeritud vastused.

Mõne aja pärast asus autor Kinnistusraamatu dokumentatsiooni abil päris registrit integreerima. Kui varem tagastati simuleeritud vastus, siis nüüd tuli iga 6 põhimeetodi sees kutsuda välja uus abimeetod, mis teeb päringu Kinnistusraamatusse ja tagastab vastuse XML-formaadis. Kirjutati ja rakendati ka uus abimeetod, mis selle XML-vastuse

JSON-formaati teisendab. Viimaks lisati autoriseerimise funktsionaalsus, mis välistab ilma õiget API võtit sisestamata *endpoint*i kasutamise. Seega ei saa Hoovi nimel päringuid teha ja ettevõttele sellega kulutusi tekitada.

## 3.2 Ehitisregister

Mikroteenus loodi DynamoDBsse salvestatud Ehitisregistri kirjete pärimiseks. Seda kirjeldab Joonis 3.


Joonis 3. Ehitisregister.

Ette oli nähtud ainult üks *endpoint:*

*GET: buildings/{building-ID}/{part-ID}*

Hoone ID (*building-ID*) pidi olema kohustuslik ning osa ID (*part-ID*) valikuline. Vastus tuli tagastada JSON-formaadis. Kaks välja olid salvestatud DynamoDB-le eripärases *Map*-formaadis ning need tuli JSON-i jaoks lihtsustada. Kui *part-ID* oli puudu, tuli järgida kindlat algoritmi:

- Lugeda kokku objektid õige *building-ID*ga ning *part-ID*ga, mis on suurem kui 0
    - Kui on ainult üks selline kirje, tagastada see
    - Vastasel korral sorteerida välja *osa_id* järgi ning
        - Tagastada esimene, millel "osa_liik" == "K"
        - Kui selliseid pole, tagastada esimene kirje
- Kui üldse sobivaid kirjeid pole, tagastada veateade

Peale Lambda-funktsiooni üldstruktuuri loomist kirjutati kaks meetodit: ühe URLis sisaldub *part-ID* ja teise omas mitte. Üldine loogika nende sees oli järgnev

1. Küsida andmebaasist kirjed

    1.1. Teha päring DynamoDB-sse

2. Lugeda kokku tagastatud kirjete arv
3. Kui kirjeid on üle 1, filtreerida välja õige kirje (vastavalt algoritmile)
4. Teisendada kaks keerulises formaadis välja
5. Tagasta vastus

Põhjalikumat kirjeldust väärib keerulisemate väljade teisendamine. DynamoDB-s on kaks välja: „hoone_tehnilised andmed“ ja „osa_tehnilised_andmed“. Baasist tagastatakse väljad järgneval kujul:

```
"osa_tehnilised_andmed": {"TEHNO_ELPLIIT": [{"1901": {"S": "puudub"}}]}
```

Vaja oli aga lihtsamat kuju:

```
"osa_tehnilised_andmed": {"TEHNO_ELPLIIT": [["1901", puudub"]]}
```

Lihtsustamiseks tuli kirjutada mõned read koodi (Joonis 4).

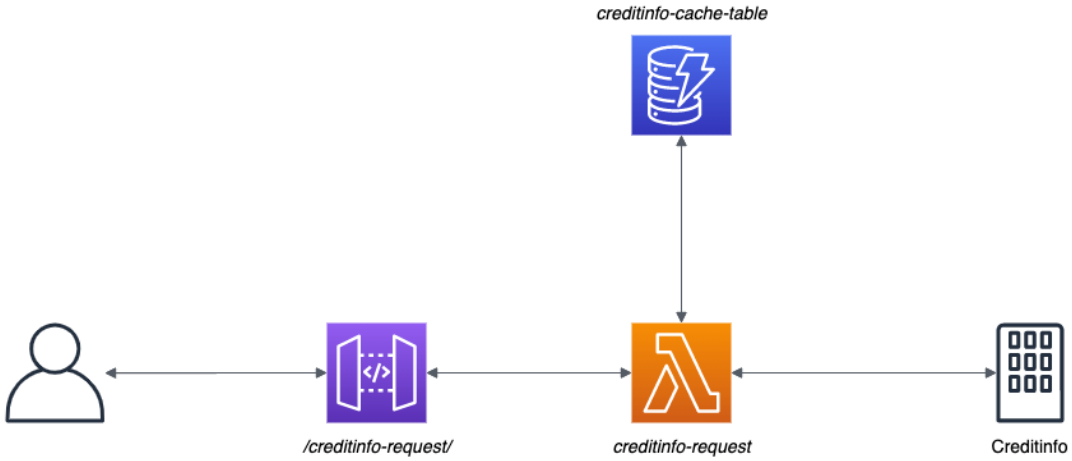
```
part_technical_data = result['osa_tehnilised_andmed']
    transformed_part_technical_data = {}
    for item in part_technical_data:
        outer_map = part_technical_data[item][0]
        outer_key = list(part_technical_data[item][0].keys())[0]
        inner_map = outer_map[outer_key]
        inner_key = list(inner_map.keys())[0]
        inner_value = inner_map[inner_key]
        new_list = [[outer_key, inner_value]]
        new_dict_entry = {
            item: new_list
        }
        transformed_part_technical_data.update(new_dict_entry)
    result['osa_tehnilised_andmed'] = transformed_part_technical_data
```

Joonis 4. Kood Ehitisregistri andmete lihtsustamiseks.

Arenduse tulemus oli positiivne: mikroteenus tagastas andmed nõutud kujul.

## 3.3 Creditinfo

Integratsioon Creditinfo registriga oli osaliselt juba valmis kirjutatud teiste ettevõtte töötajate poolt. Autori ülesandeks oli REST *endpoint*i loomine, mis laseks Creditinfost tulnud XML-vastuse õigesse formaati teisendada ning salvestaks selle andmebaasi. Üldist toimimismehhanismi näitab Joonis 5.




Joonis 5. Creditinfo.

Ainsa (GET) päringu URL on:

*/privatePerson/{id}/{teenuse_kood}*

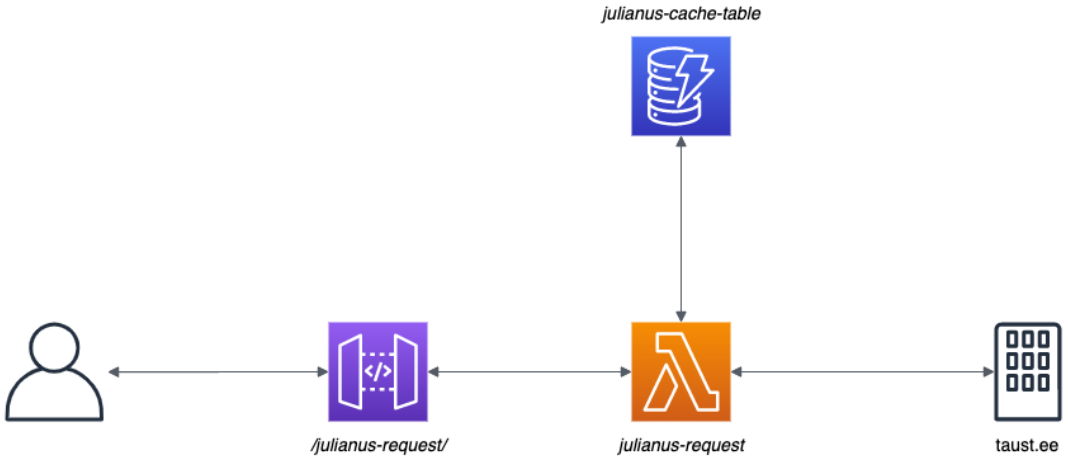
Algoritm on järgnev:

1. kontrollitakse, kas URL-i lõpus on õige API võti (turvakaalutlustel);
2. kontrollitakse, kas isikukood on kehtiv;
3. kontrollitakse, kas DynamoDB-s on viimase 3 päeva jooksul lisatud kirjeid; kui on, siis tagastatakse need;
4. vastasel korral tehakse SOAP-päring Creditinfosse;
5. XML-vastus teisendatakse JSON-formaati;
6. Mõlemas formaadis vastused salvestatakse DynamoDB-sse
7. JSON-vastus tagastatakse kasutajale.

Autor kirjutas meetodid DynamoDB-st vastuse küsimiseks ja sinna salvestamiseks, kuid mitte XML-vastuse teisendamiseks ega isikukoodi kontrollimiseks; viimased oli teiste töötajate poolt juba valmis kirjutatud.

## 3.4 Taust.ee

Taust.ee mikroteenus tuli tööülesande kirjelduse põhjal luua Creditinfo integratsiooni eeskujul. Üldine tööpõhimõtet kujutab Joonis 6. Algoritm, mis oli toodud Creditinfo alampeatükis, kehtib sarnaselt ka taust.ee puhul. Lisaks pidi autor kirjutama koodi SOAP-päringu tegemiseks kui ka XML-vastuse töötlemiseks.




Joonis 6. Taust.ee.

Olulisemat osa päringu tegemise meetodist demonstreerib Joonis 7.

```
session = Session()
client = Client(SOAP_WSDL, transport=Transport(session=session))
log_info(f"Trying to get report with personal code {personal_code}")
customer_data = {
    "username": USERNAME,
    "password": PASSWORD,
    "code": personal_code
}
result = client.service.getByCode(customer_data)
```

Joonis 7. Kood päringu tegemiseks taust.ee-sse.

*Requests*-mooduli abil luuakse *Session*-objekt. *Zeep*-mooduli abil luuakse *Transport*-objekt, antakse sellele parameetrina sessioon, luuakse *Client*-objekt ning antakse sellele omakorda parameetrina *.wsdl*-fail ja *Transport*-objekt. Zeepi dokumentatsiooni järgi läheb vaja *Session*-objekti ja *Transport*-objekti päringutele lisatavate parameetrite korduvkasutamiseks [46]. *Client*-objekti kaudu pääsetakse ligi *.wsdl* failis kirjeldatud teenustele. *USERNAME* ja *PASSWORD* on olemas keskkonnamuutujatena.

XML-vastuse töötlemine on teostatud kahes meetodis, nendest esimese põhilist rida näitab Joonis 8:

```
parsed_claims = parse_claims(result['_value_1'][1])
```

Joonis 8. Rida koodi taust.ee vastusest sisulise osa kättesaamiseks.

Vastusest küsitakse selle sisukas osa ning edastatakse see teisele meetodile. Teise meetodi põhilist sisu demonstreerib Joonis 9:

```
claim_list = []
for claim in claims:
    claim = claim['_value_1']
    claim_object = {
        "claimant": claim[0],
        "startDate": claim[1],
        "endDate": claim[2],
        "amount": claim[3],
        "status": claim[4],
        "activityArea": claim[5],
    }
    claim_list.append(claim_object)
```

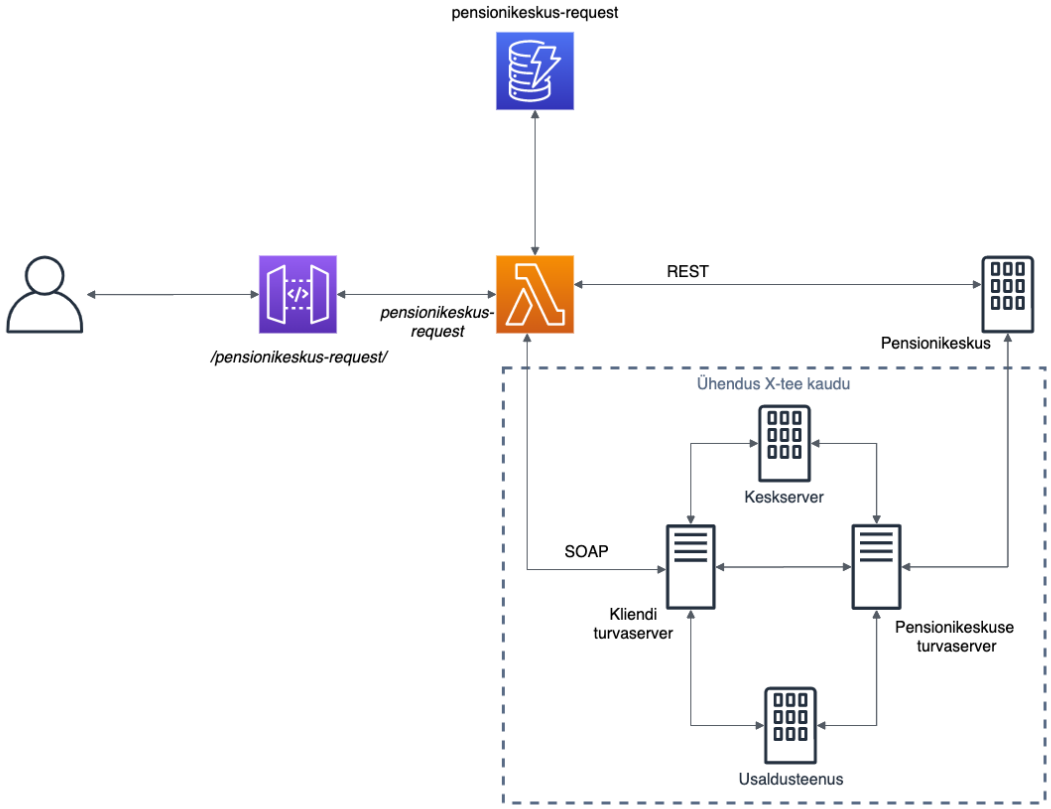
Joonis 9. Kood taust.ee tagastatud nõuete listi teisendamiseks lihtsamale kujule.

*Claim-list*-i lisatakse kõik võlanõuded JSON-i objekti kujul; algselt on need *list*i kujul ja seega raskemini loetavad.

## 3.5 Pensionikeskus

Pensionikeskuse integratsiooni põhiosa oli juba varem firma teiste töötajate poolt loodud, ent autorile määratud tööülesanne nägi ette REST *endpoint*-i loomise. See pidi tööülesande kirjelduse järgi tagastama info isiku kuusissetuleku kohta (kui selline info saadaval on).

Mikroteenuse tööd iseloomustab üldisel kujul Joonis 10.




Joonis 10. Pensionikeskus.

Pensionikeskusest saab infot küsida tegelikult kahes eri vormis:

1. Tasuta REST päringuga, mis tagastab vastuse selle kohta, kas isikul on aktiivne pensionikonto ning kas 2020. aasta muudatuste järel on sissemaksed peatatud;
2. X-tee kaudu tasulise päringuga, kust saab küsida teavet isiku pensionikontole tehtud sissemaksetest.

Tööülesandes oli ette nähtud ainult üks päring:

*GET: /pensionikeskus-request/{ID}?preCheckOnly=1*

ID asemel tuleb sisestada isikukood. *preCheckOnly* on valikuline parameeter: kui selle väärtuseks on 1, siis on eesmärgiks saada ainult teada, kas inimesel on aktiivne pensionikonto; vastasel korral on eesmärgiks saada ka sissetuleku info.

*Endpoint*-i kood kirjutati järgneva üldise algoritmi järgi:

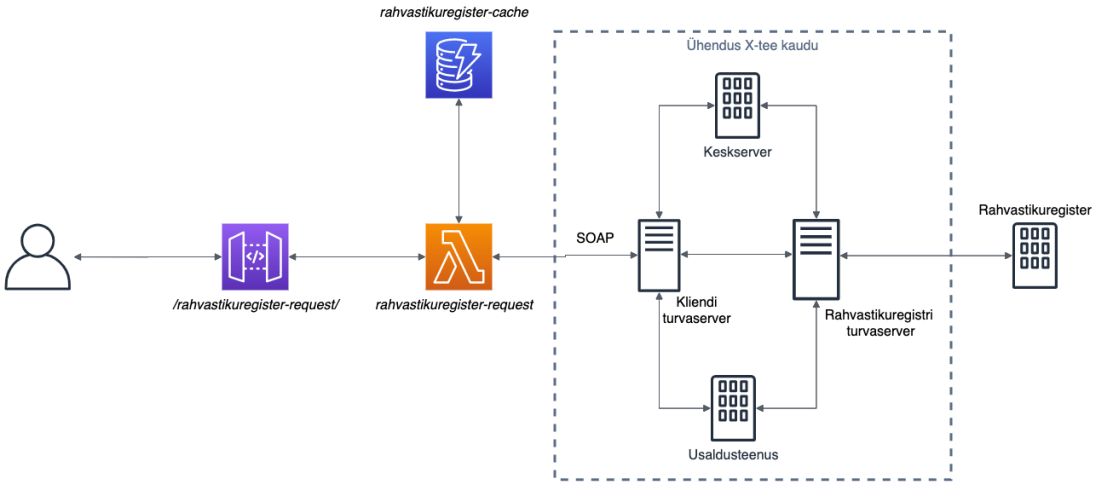
1. kontrollitakse, kas URL-i lõpus on õige API võti (turvakaalutlustel);
2. tehakse tasuta eelpäring;
    2.1. kui vastus on negatiivne (isikul puudub aktiivne pensionikonto); tagastatakse vastav teade;
    2.2. kui vastus on positiivne (isikul on aktiivne pensionikonto) JA *preCheckOnly == 1,* tagastatakse vastav teade;
    2.3. kui vastus on positiivne (isikul on aktiivne pensionikonto) JA *preCheckOnly != 1*, jätkatakse protsessi;
3. Kontrollitakse, kas DynamoDB-s on hiljuti salvestatud vastuseid;
    3.1. Kui on, tagastatakse need;
    3.2. Kui pole, jätkatakse protsessi;
4. Tehakse X-tee päring Pensionikeskusse;
5. Töödeldakse vastust ja arvutatakse välja keskmine kuusissetulek;
6. Salvestatakse päringu vastus ja kuusissetulek DynamoDB-sse;
7. Tagastatakse järgnevas stiilis JSON-vastus:

```
{
    "calculatedGrossIncome": <kuusissetulek>,
    "originalResponse": <JSON-formaadis Pensionikeskuse vastus>
}
```

Autori kirjutas meetodid eelpäringu tegemiseks, vastuste salvestamiseks DynamoDB-sse, keskmise kuusissetuleku arvutamiseks, pensionikonto sissemaksete jätkumise kontrolliks 2020. aasta muudatuste järel; osa funktsionaalsusest (X-teega suhtlemine) oli ettevõtte teiste töötajate poolt juba valmis kirjutatud.

## 3.6 Rahvastikuregister

Suhtlus Rahvastikuregistriga käib X-tee kaudu, mistõttu sai olulisel määral taaskasutada Pensionikeskuse integratsiooni koodi. Rakenduse tööd kirjeldab Joonis 11.




Joonis 11. Rahvastikuregister.

Tööülesande kirjelduses oli ette nähtud kaks päringut:

1) *GET: /rahvastikuregister-request/RR404/{ID}* (teovõime)
2) *GET: /rahvastikuregister-request/RR437/{ID}* (kas isik on elus)

Kummagi päringu jaoks mõeldud meetodi sisse kirjutati sarnase algoritmiga kood:

1. Kontrollida, kas saadeti õige API võti (turvakaalutlustel)
2. Kontrollida, kas DynamoDB andmebaasis on värskeid kirjeid, mida taaskasutada saaks
   2.1. kui on, tagastada see kirje;
   2.2. kui pole, jätkata;
3. teha X-tee päring Rahvastikuregistrisse;
4. salvestada vastus andmebaasi;
5. tagastada vastus;

Kummagi päringu jaoks oli ka eraldi meetod, mis punktis 3) mainitud päringut Rahvastikuregistrisse teeb. Need meetodid kirjutati Pensionikeskuse rakenduses olnud meetodite eeskujul. Nende meetodite tööalgoritm on üldjoontes järgnev:

1. küsida keskkonnamuutujad (seal on kirjas serveri URL, kasutaja andmed, teenuse parameetrid jms);
2. Luua *XroadClient* objekt (sellesse objekti sisestatakse eelmises punktis mainitud muutujad ning selle objekti kaudu pääseb vajalikele teenustele ligi);
3. *XroadClient*-i objekti vastava meetodi abil luua *Client*-objekt (sarnaselt Creditinfo alampeatükis kirjeldatuga);
4. *Client*-objekti vastava meetodi abil luua päised;
5. C*lient*-objekti kaudu käivitada soovitud teenus (RR404 või RR437), andes kaasa päringu parameetrid (inimese isikukood) ja päised;
6. Teisendada saadud SOAP-vastus JSON (sõnastiku) formaati.

# 4 Analüüs

## 4.1 Väljakutsed

API loomise käiguse esines raskusi, mis projekti aeganõudvamaks tegid.

Paar pudelikaela tuli ette Kinnistusraamatu integratsiooni kirjutades. Et päringuid saaks teatud aja vältel kulude kokkuhoidmiseks taaskasutada, tuli need salvestada AWS DynamoDB andmebaasi. Probleem oli selles, et DynamoDBs pole eraldi kuupäeva salvestamiseks mõeldud andmetüüpi, mistõttu oli algul ebaselge, kuidas sealt näiteks viimase 7 päeva jooksul salvestatud kirjeid küsida saaks. Siin sai veebisaidil medium.com avaldatud Kathy Danielsi artikli abil selgeks, et sellele probleemile oli mõeldud ning lahenduseks on kuupäeva salvestamine stringina [47]. Seejuures oli oluline, et kuupäev oleks salvestatud ISO 8601 standardi järgi ehk formaadis YYYY-MM-DD, kus YYYY on aasta, MM on kuu ja DD on päev [48]. See kuupäev tuli Danielsi kirjeldusel määrata sekundaarvõtmeks [47]. Kui teha andmebaasi päring, kus primaarvõtmena sisestatakse inimese isikukood ning lisaks antakse võrratuse vormis sekundaarvõtmena kuupäev (näiteks *2019-12-01* < kuupäev < *2020-01-01),* siis ongi võimalik küsida andmebaasist kindla ajavahemiku kirjeid.

Taust.ee puhul oli kõige keerulisem SOAP-päringu tööle saamine ning vastuse teisendamine. Nimelt Creditinfo vastavas meetodis, mida eeskujuna kasutati, toimus autentimine *Session*-objekti abil (vt Joonis 12).

```
session = Session()
credentials = get_secret(CREDENTIALS)
session.auth = HTTPBasicAuth(credentials.get('creditinfo_username'),
credentials.get('creditinfo_password'))
client = Client(SOAP_WSDL, transport=Transport(session=session))
result = client.service.getReport(XML_REPORT_CODE, personal_code)
```

Joonis 12. Autentimine Creditinfo mikroteenuses.

Taust.ee puhul tuli aga kasutajanimi ja parool sõnastikuna päringuga kaasa anda (vt Joonis 13).

```
customer_data = {
    "username": USERNAME,
    "password": PASSWORD,
    "code": personal_code
}
result = client.service.getByCode(customer_data)
```

Joonis 13. Autentimine taust.ee mikroteenuses.

Selle probleemi ja taust.ee alampeatükis kirjeldatud vastuse töötlemise probleemi lahendamisel oli abi ennekõike taust.ee dokumentatsioonist [29] ning testimise järel logide uurimisest.

Pensionikeskuse integreerimisel tekkis väike takistus sellest, kui JSON-vastus jäeti teisendamata stringiks meetodiga *json.dumps(),* mistõttu vastus salvestus andmebaasi ebakorrektsel kujul.

Rahvastikuregistri integreerimisel oli mitu pudelikaela. Esiteks tuli paika saada päringu parameetrid. Eeskujuna kasutatud Pensionikeskuse rakendusel oli isikukood antud eraldi muutujana sulgude sees (vt Joonis 14).

```
resp = client.service.laekumised_maksuametist(ISIKUKOOD=id_code,
_soapheaders=header)
```

Joonis 14. Isikukoodi edastamine Pensionikeskusse.

Rahvastikuregistri puhul tuli aga luua eraldi sõnastik, parameeter sinna sisse panna ja see sõnastik siis omakorda parameetrina teenusele ette anda (vt Joonis 15).

```
request = {
    'Isikukood': id
}
resp = client.service.RR437(request=request, _soapheaders=header)
```

Joonis 15. Isikukoodi edastamine Rahvastikuregistrisse.

Parameetri paika saamisel tuli lähtuda .wsdl failist. Kui .wsdl faili oli ka õige serveri URL lisatud, siis sai asuda rakendust testima. Päring RR437 töötas, ent päring RR404 saatis tühja vastuse. Ettevõtte järelepärimise peale vastas registri pidaja, et päring RR404 saadabki tühja vastuse, kui inimene on teovõimeline, ning vastus sisaldab infot vaid siis, kui teovõime on piiratud. See oli viimane suurem tõrge Rahvastikuregistri integratsiooni arendamisel.

## 4.2 Alternatiivid

Integratsioon oli kirjutatud võimalikult lihtsalt ning sellist funktsionaalsust, mida ettevõte ei nõudnud, koodi ka ei lisatud. Samas oleks teoreetiliselt olnud võimalik lahendus kirjutada teises programmeerimiskeeles või paigaldada see mõnda teise pilvekeskkonda.

AWS lubab kasutada järgmisi programmeerimiskeeli [49]:

- Java
- Javascript
- C#
- C++
- PHP
- Python
- Ruby
- Go

Kui oletada, et firma ülejäänud infosüsteemis kasutavad keeled ei piiranuks ega mõjutanuks üldse antud integratsiooni arendamist, siis võib arutleda, kuidas mõni teine keelevalik mõjutanuks arendust. Välistada võib Go, Ruby ja C++, kuna autor pole neid õppinud. Java või C# kasutamine teeks koodi tõenäoliselt pikemaks ja keerulisemaks, kuna süntaks on Pythoniga võrreldes teistsugune (näiteks need keeled on rangelt määratud andmetüüpidega, samuti tuleb Javas ja C#-s kasutada looksulge meetodite ja klasside piiritlemiseks). Samas jälle teeksid need keeled arenduse mõnes mõttes ka lihtsamaks, kuna erinevalt Pythonist on need kompileeritavad keeled. Seega näitaks juba IDE ette peamised probleemid ning need saab ilma testimata ära parandada. Javascripti kasutamine teeks *front-end*i ja *back-end*i omavahel sarnasemaks, kuna *front-end*is kasutatakse niikuinii JS-i. PHP-ga on mainitud keeltest autoril kõige vähem kogemusi, kuid ilmselt annaks ka seda integratsiooni arendamiseks kasutada.

Veebilehe cloudhealth.com andmetel on AWS-i peamisteks konkurentideks Microsoft Azure, Google Cloud, Oracle Cloud, IBM Cloud ja Alibaba Cloud [50]. Kuna tuntumad on Microsofti ja Google-i teenused, võib põhjalikumalt analüüsida just neid. AWS-il on cloudhealth.com-i väitel ülekaalukalt kõige suurem turuosa ning kõige rohkem funktsionaalsust; samas on see ka tuntud kasutuskeerukuse poolest [50]. Sama portaali andmeil loetakse Microsoft Azure'i eelisteks just lihtsust, eriti Microsoft raamistike nagu

.NET kasutajatele, ning erinevalt AWS-ist hübriid-pilvelahenduste lubamist (rakenduse osad komponendid võivad paikneda lokaalselt või mõne teise teenusepakkuja juures) [50]. Google Cloudi eeliseks peab cloudhealth.com pilvekonteinerteenuse taset ja hindu ning masinõppe arengutaset [50]. Peamiseks funktsionaalsuseks, mida autoril tarvis oli AWS-is, oli AWS Lambda (serverivaba arvutusteenus). Selle analoogiks Microsofti puhul on Eyal Poseneri hinnangul Azure Functions ning Google'i puhul Cloud Functions [51]. Teiseks oluliseks funktsionaalsuseks oli DynamoDB, mille Microsofti vasteks portaali trustradius.com andmeil oli Azure Cosmos DB ning Google'i vasteks Google Cloud Datastore [52]. Seega töös käsitletava rakenduse loomiseks vajalik funktsionaalsus oli kõigil kolmel teenusepakkujal olemas ning ühegi teenusepakkuja eelistamiseks teistele polnud tugevat põhjust, vaid AWS-i kasutamise otsus lähtus puhtalt firma varasemast kasutuskogemusest ja -harjumusest.

## 4.3 Äriprotsess

Andmete teekonna äriprotsessis võib üldjoontes jagada kahte etappi: registreerimisvormi täitmine ning raporti koostamine.

Potentsiaalne klient täidab registreerimisvormi, kusjuures info kättesaamiseks ei pea vormi isegi ära saatma. Registreerimisvormi algul luuakse iga kliendi jaoks UUID. Kui kasutaja klõpsab mõne sisestamisvälja või nupu peale, saadetakse vorm koos seni sisestada jõutud andmetega eraldi AWS Lambda funktsiooni, mille ainuke roll on andmete logimine AWS CloudWatchi. Kui kliendil näiteks jookseb internetibrauser viimasel lehel kokku ja ära saatmine ei õnnestu, saab klient pöörduda Hoovi poole ning firma töötaja saab CloudWatchist kontrollida, kas vajalikud andmed on olemas. Sel juhul pole vormi uuesti täitmine vajalik.

Andmete teekonna teine etapp põhineb AWS *Step Functionit*el. AWS-i dokumentatsioon kirjeldab, et kõik sammud äriprotsessis on koondatud olekumasinaks (*state machine)* ning need sammud on omavahel kindlas järjekorras, kusjuures võib olla mitu varianti, mis samm eelmisele sammule järgneb; iga sammu tulemuseks on mingi olek (*state*) [53]. See, milline samm järgmisena käivitatakse, sõltub eelmise sammu tulemusest ehk olekust. Olek antakse järgmisele sammule sisendiks ning see teeb sisendiga oma protsessi ära ja annab selle omakorda järgmisele sammule edasi. Vormi lõpetamisel saadetakse andmed POST-päringuga olekumasinasse, mis küsib vajalikud andmed käesolevas töös kirjeldatud mikroteenustelt ning salvestab kõik andmed Pipedrive'i, kustkaudu need muutuvad firma töötajatele nähtavaks. Sellisel teel saab ettevõte loetud sekunditega ülevaate vajalikest andmetest ning teeb otsuse, kas taotleja saab krediiti või ei saa.

## 4.4 Edasised arendusplaanid

Integratsioonide arendamine polnud Hoovis lõputöö kirjutamise ajal lõppenud. Tulevikus on plaanis Eesti integratsioonide edasiarendus ning välistatud pole välismaiste integratsioonide arendus.

Esiteks võetakse juba integreeritud registritest kasutusele uued alamteenused. Näiteks on plaanis integreerida Creditinfo alamteenused, mis annavad infot juriidiliste isikute

võlgnevuste kohta, kuna seni saadi sealt infot ainult eraisikute kohta. Samuti on plaanis võtta kasutusele täiendatud makseteenuste direktiiviga (tuntud ka kui PSD2 – *Payment Service Directive 2*) pakutav kontoteabe teenus (AIS – *Account Information Service*), mis Swedbanki ja PocoPay sõnul võimaldab vastava API kaudu ettevõttel küsida kliendi pangakonto viimase kahe aasta väljavõtet [54] [55]. See oleks täiendav viis veenduda üürigarantii või ärilaenu taotleja krediidivõimekuses.

Teiseks on välisturule laienemise korral võimalik välisriikide registrite integreerimine. Kõige tõenäolisem on integratsioon Läti registritega. Näiteks Läti rahvastikuregistrist on võimalik küsida teavet isiku teovõime kohta, maaregistrist aga infot kinnistute kohta. Samuti saab Läti äriregistrist küsida infot ettevõttega seotud isikute kohta.

Kui Eestiga seotud edasiarendused on kindlad, siis teiste riikidega seotud arendused sõltuvad ettevõtte edasistest äriplaanidest.

## 4.5 Isikuandmete kaitse

Eestis Vabariigis reguleerib isikuandmete töötlemist, täpsemalt füüsiliste isikute isikuandmete automatiseeritud töötlemist, Euroopa Parlamendi ja Nõukogu määrus 2016/679 ehk isikuandmete kaitse üldmäärus (tuntud ka ingliskeelse lühendi GDPR (*General Data Protection Regulation*) järgi) ning selle alusel täiendatud riigisisene isikuandmete kaitse seadus. Viimati mainitud seadusest lähtub ka Hoovi privaatsuspoliitika, mis tagab ettevõtte klientide isikuandmete privaatsuse [56].

Andmekaitse Inspektsiooni sõnul peab kõigi isikuandmete kogumiseks olema mingi konkreetne eesmärk [57]. Järgnevalt on kategoriseeritud kujul toodud andmed, mida Hoovi kogub, ning nende kogumise põhjus Hoovi privaatsuspoliitika järgi [56]:

1. Põhilised isikuandmed ja kontaktandmed (näiteks isikukood, e-posti aadress): need on vajalikud, et teha päringuid välistesse registritesse, tuvastada pettust ja identiteedivargust ning teavitada kliente muutustest Hoovi teenustes
2. Juriidiline staatus Eesti Vabariigis: Hoovi ei sõlmi lepingut isikutega, kes pole Eesti kodanikud ega residendid
3. Info teovõimelisuse kohta: Hoovi ei sõlmi lepingut teovõimetute isikutega
4. Andmed rahalise võimekuse, kinnistute, maksehäirete ja ametlike teadaannete kohta: on vajalikud finantsvõimekuse hindamiseks üürilepingus võetud kohustuste täitmiseks ning maksehäire tekkimise tõenäosuse arvutamiseks üürimaksete tasumisel
5. Küpsised ja digitaalsed kasutusandmed: on vajalikud teenuse pakkumiseks ja haldamiseks, selle toimimise ja kättesaadavuse parandamiseks, kasutamise jälgimiseks ning tehniliste tõrgete eemaldamiseks
6. Sotsiaalmeediakontode olemasolu ja sisu: on vajalikud finantsvõimekuse hindamiseks ning maksehäirete tekkimise tõenäosuse arvutamiseks

Hoovi kogub füüsilise isiku kohta ainult tavalisi isikuandmeid. Andmekaitse Inspektsiooni sõnul on tavalised isikuandmed need isikuandmed, mis ei kuulu tundlike ega eriliiki isikuandmete hulka, ent mille järgi on siiski võimalik teda otseselt või kaudselt tuvastada [58]. Õigusliku aluse nende andmete kogumiseks annab Hoovile seadusejärgne kohustus veenduda laenuvõtja usaldusväärsuses [59]. Nimelt on krediidiandja vastavalt Võlaõigusseaduse §403[4] kohustatud „järgima vastutustundliku laenaja põhimõtet“ [60]. Selle käigus tuleb omandada teave, „mis võimaldab hinnata, kas tarbija on võimeline

krediidi lepingu lepingus kokkulepitud tingimustel tagasi maksma“, ning selle alusel hinnata tarbija krediidivõimelisust [60].

Klientide andmeid jagatakse töös kirjeldatud väliste registripidajatega ehk Creditinfo, Krediidiregistri (taust.ee pidaja), Pensionikeskuse, Rahvastikuregistri ja Kinnistusraamatuga, samuti AWS-iga, kus andmeid majutatakse. Mainitud registripidajad asuvad Eestis ning andmed majutatakse AWS-i Euroopa Liidus asuvatesse serveritesse, seega Andmekaitse Inspektsiooni sõnul andmete töötlemisele täiendavaid nõudeid ei ole [61].

Hoovi säilitab andmeid piiratud aja jooksul. Kui isik on esitanud taotluse, ent lepingut sõlminud pole, siis andmed kustutatakse 2 aasta jooksul pärast taotluse esitamist [56]. Lepingu sõlmimise korral säilitatakse andmeid lepingu kestvuse ajal ning pärast lepingu lõppemist veel kuni 10 aastat [56].

Kliendil on oma andmete suhtes kindlad õigused [56]:

- Saada teada, milliseid andmeid tema kohta Hoovil on
- Nõuda andmete parandamist
- Nõuda andmete kustutamist
- Nõuda andmete töötlemise piiramist ja lõpetamist

Seega Hoovi tagab oma klientide isikuandmete kaitse ning kasutab neid vastustundlikult.

# 5 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli luua REST API, mis teisendaks välistest registritest saadavad andmed ettevõtte Hoovi äriprotsessi jaoks sobivasse formaati. Töö käigus loodi mikroteenused Kinnistusraamatule, Rahvastikuregistrile, taust.ee-le ja Ehitisregistrile, samuti lisati *endpoint* juba olemas olnud Creditinfo ja Pensionikeskuse rakendustele. Loodi funktsionaalsus, mis küsib registrist andmeid, teisendab XML-vastused JSON-formaati, lihtsustab JSON-vastuseid, salvestab vastused ettevõtte andmebaasi ning tagastab need kasutajale. Kõikide registrite integreerimine lõppes positiivse tulemusega. Mikroteenuseid kasutatakse äriprotsessis selleks, et hinnata krediidi taotlejate maksevõimet ning otsustada, kas krediiditaotlus vastu võtta või tagasi lükata.

# Kasutatud kirjandus


[1] AWS, „What is Amazon API Gateway?,“ [Võrgumaterjal]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html. [Kasutatud 22. veebruar 2021].

[2] „Cambridge Dictionary,“ [Võrgumaterjal]. Available: https://dictionary.cambridge.org/. [Kasutatud 22. veebruar 2021].

[3] ProfuctPlan, „Backlog,“ [Võrgumaterjal]. Available: https://www.productplan.com/glossary/backlog/. [Kasutatud 22. veebruar 2021].

[4] AWS, „Amazon CloudWatch,“ [Võrgumaterjal]. Available: https://aws.amazon.com/cloudwatch/. [Kasutatud 22. veebruar 2021].

[5] AWS, „What Is Amazon DynamoDB?,“ [Võrgumaterjal]. Available: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html. [Kasutatud 22. veebruar 2021].

[6] SmartBear, „API Endpoints - What Are They? Why Do They Matter?,“ [Võrgumaterjal]. Available: https://smartbear.com/learn/performance-monitoring/api-endpoints/. [Kasutatud 22. veebruar 2021].

[7] „Git,“ [Võrgumaterjal]. Available: https://git-scm.com/. [Kasutatud 22. veebruar 2021].

[8] TechTerms, „JSON,“ [Võrgumaterjal]. Available: https://techterms.com/definition/json. [Kasutatud 22. veebruar 2021].

[9] AWS, „Lambda,“ [Võrgumaterjal]. Available: https://aws.amazon.com/lambda/. [Kasutatud 22. veebruar 2021].

[10] Branch, „Query Parameters,“ [Võrgumaterjal]. Available: https://branch.io/glossary/query-parameters/. [Kasutatud 22. veebruar 2021].

[11] Codecademy, „What is REST?,“ [Võrgumaterjal]. Available: https://www.codecademy.com/articles/what-is-rest. [Kasutatud 22. veebruar 2021].

[12] GeeksforGeeks, „Service-Oriented Architecture,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/service-oriented-architecture/. [Kasutatud 22. märts 2021].

[13] TutorialsPoint, „What is SOAP?,“ [Võrgumaterjal]. Available: https://www.tutorialspoint.com/soap/what_is_soap.htm. [Kasutatud 22. veebruar 2021].

[14] E. Christensen, F. Curbera, G. Meredith ja S. Weerawarana, „Web Services Description Language (WSDL) 1.1,“ [Võrgumaterjal]. Available: https://www.w3.org/TR/wsdl.html. [Kasutatud 22. veebruar 2021].

[15] TechTerms, „XML,“ [Võrgumaterjal]. Available: https://techterms.com/definition/xml. [Kasutatud 22. veebruar 2021].

[16] Riigi Infosüsteemide Amet, „Andmevahetuskiht X-tee,“ [Võrgumaterjal]. Available: https://www.ria.ee/et/riigi-infosusteem/andmevahetuskiht-x-tee.html. [Kasutatud 22. veebruar 2021].

[17] C. Richardson, „Microservices Architecture,“ [Võrgumaterjal]. Available: https://microservices.io/. [Kasutatud 22. veebruar 2021].

[18] X. S. I. C.-P. R. &. E. C. Larrucea, "Microservices," *IEEE Software,* no. 35, pp. 96-100, 2018.

[19] A. Veldre, „Sissejuhatus X-teesse (osa 3),“ [Võrgumaterjal]. Available: https://blog.ria.ee/sissejuhatus-x-teesse-osa-3/. [Kasutatud 10. mai 2021].

[20] A. Veldre, „Sissejuhatus X-teesse (osa 2),“ 28. august 2015. [Võrgumaterjal]. Available: https://blog.ria.ee/sissejuhatus-x-teesse-osa-2/. [Kasutatud 9. mai 2021].

[21] A. Veldre, „Sissejuhatus X-teesse (osa 1),“ [Võrgumaterjal]. Available: https://blog.ria.ee/sissejuhatus-x-teesse-osa-1/. [Kasutatud 10. mai 2021].

[22] RIA, „Andmevahetuse kirjeldus,“ [Võrgumaterjal]. Available: https://abi.ria.ee/xtee/et/x-tee-juhend/kuidas-kasutada-x-teed/andmevahetuse-kirjeldus. [Kasutatud 9. mai 2021].

[23] RIK, „e-kinnistusraamat,“ [Võrgumaterjal]. Available: https://www.rik.ee/et/e-kinnistusraamat. [Kasutatud 20. veebruar 2021].

[24] Justiitsministeerium, „Kinnistusraamat,“ [Võrgumaterjal]. Available: https://www.just.ee/et/eesmargid-tegevused/kinnisvara-abieluvara-parimine/kinnistusraamat. [Kasutatud 20. veebruar 2021].

[25] RIK, „Kinnistusraamatu XML teenus,“ [Võrgumaterjal]. Available: https://www.rik.ee/sites/www.rik.ee/files/elfinder/article_files/kinnistusraamatu_xml_teenus_02.07.2019.docx. [Kasutatud 20. veebruar 2021].

[26] „Ehitisregister,“ [Võrgumaterjal]. Available: https://www.ehr.ee/. [Kasutatud 20. veebruar 2021].

[27] Ehitisregister, „Ehitisregistri avaandmete veebiteenused,“ [Võrgumaterjal]. Available: https://avaandmed.ehr.ee/. [Kasutatud 20. aprill 2021].

[28] taust.ee, „Firmast,“ [Võrgumaterjal]. Available: https://www.taust.ee/firmast. [Kasutatud 20. veebruar 2021].

[29] taust.ee, „Exporter SOAP WebService interface description,“ [Võrgumaterjal]. Available: https://test.taust.ee/export/wsdl-claim.php. [Kasutatud 21. veebruar 2021].

[30] „Creditinfo,“ [Võrgumaterjal]. Available: https://www.creditinfo.ee/. [Kasutatud 20. veebruar 2021].

[31] Siseministeerium, „Rahvastikuregister,“ [Võrgumaterjal]. Available: https://www.siseministeerium.ee/et/eesmark-tegevused/rahvastikutoimingud/rahvastikuregister. [Kasutatud 20. veebruar 2021].

[32] Nasdaq, „Pensionikeskus,“ [Võrgumaterjal]. [Kasutatud 20. veebruar 2021].

[33] Statistics Times, „Top Computer Languages,“ [Võrgumaterjal]. Available: https://statisticstimes.com/tech/top-computer-languages.php. [Kasutatud 20. veebruar 2021].

[34] w3schools, „Python Introduction,“ [Võrgumaterjal]. Available: https://www.w3schools.com/python/python_intro.asp. [Kasutatud 20. veebruar 2021].

[35] Atlassian Bitbucket, „A brief overview of Bitbucket,“ [Võrgumaterjal]. Available: https://bitbucket.org/product/guides/getting-started/overview. [Kasutatud 20. veebruar 2021].

[36] G2, „Bitbucket Alternatives & Competitors,“ [Võrgumaterjal]. Available: https://www.g2.com/products/bitbucket/competitors/alternatives. [Kasutatud 20. veebruar 2021].

[37] Atlassian Bitbucket, „Pricing,“ [Võrgumaterjal]. Available: https://www.atlassian.com/software/bitbucket/pricing. [Kasutatud 20. veebruar 2021].

[38] GitHub, „Pricing,“ [Võrgumaterjal]. Available: https://github.com/pricing. [Kasutatud 20. veebruar 2021].

[39] GitHub Blog, „New year, new GitHub: Announcing unlimited free private repos and unified Enterprise offering,“ 7. jaanuar 2019. [Võrgumaterjal]. Available: https://github.blog/2019-01-07-new-year-new-github/. [Kasutatud 20. aprill 2021].

[40] GitLab, „Pricing,“ [Võrgumaterjal]. Available: https://about.gitlab.com/pricing/. [Kasutatud 20. veebruar 2021].

[41] Amazon Web Services, „Overview of Amazon Web Services,“ [Võrgumaterjal]. Available: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html. [Kasutatud 20. veebruar 2021].

[42] GitHub, „AWS Chalice,“ [Võrgumaterjal]. Available: https://github.com/aws/chalice. [Kasutatud 20. veebruar 2021].

[43] Statistikaamet, „xGate client,“ [Võrgumaterjal]. Available: https://koodivaramu.eesti.ee/statistikaamet/xgate-client. [Kasutatud 22. aprill 2021].

[44] TEHIK, „xroad-gateway,“ [Võrgumaterjal]. Available: https://koodivaramu.eesti.ee/tehik/teis/xroad-gateway. [Kasutatud 22. aprill 2021].

[45] riigiportaal-eesti.ee, „XTR,“ [Võrgumaterjal]. Available: https://koodivaramu.eesti.ee/riigiportaal-eesti.ee/xtr. [Kasutatud 22. aprill 2021].

[46] M. v. Tellingen, „Zeep: Transports,“ [Võrgumaterjal]. Available: https://docs.python-zeep.org/en/master/transport.html. [Kasutatud 20. veebruar 2021].

[47] K. Daniels, „Querying DynamoDB by Date Range,“ 12. september 2019. [Võrgumaterjal]. Available: https://medium.com/cloud-native-the-gathering/querying-dynamodb-by-date-range-899b751a6ef2. [Kasutatud 20. veebruar 2021].

[48] ISO, „ISO 8601,“ [Võrgumaterjal]. Available: https://www.iso.org/iso-8601-date-and-time-format.html. [Kasutatud 20. veebruar 2021].

[49] Amazon Web Services, „AWS Developer Center,“ [Võrgumaterjal]. Available: https://aws.amazon.com/developer/. [Kasutatud 20. veebruar 2021].

[50] CloudHealthTech, „What Do AWS Alternatives Have to Offer That AWS Can’t Provide Itself?,“ [Võrgumaterjal]. Available:

https://www.cloudhealthtech.com/blog/aws-alternatives. [Kasutatud 20. veebruar 2021].

[51] E. Posener, „7 AWS Lambda Alternatives,“ 7. detsember 2016. [Võrgumaterjal]. Available: https://www.stratoscale.com/blog/cloud/7-alternatives-aws-lambda/. [Kasutatud 20. veebruar 2021].

[52] Trustradius, „Amazon DynamoDB Competitors and Alternatives,“ [Võrgumaterjal]. Available: https://www.trustradius.com/products/amazon-dynamodb/competitors. [Kasutatud 20. veebruar 2021].

[53] Amazon Web Services, „What is AWS Step Functions?,“ [Võrgumaterjal]. Available: https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html. [Kasutatud 20. veebruar 2021].

[54] Swedbank, „PSD2 ehk täiendatud makseteenuste direktiiv,“ [Võrgumaterjal]. Available: https://www.swedbank.ee/about/about/baltic/newRegulations?language=EST. [Kasutatud 24. märts 2021].

[55] PocoPay, „Avatud Pangandus (PSD2),“ [Võrgumaterjal]. Available: https://pocopay.com/avatud-pangandus/. [Kasutatud 24. märts 2021].

[56] Hoovi, „Kasutustingimused ja privaatsuspoliitika,“ [Võrgumaterjal]. Available: https://www.hoovi.ee/privaatsuspoliitika. [Kasutatud 22. märts 2021].

[57] Andmekaitse Inspektsioon, „Andmetöötluse põhimõtted,“ [Võrgumaterjal]. Available: https://www.aki.ee/et/eraelu-kaitse/andmetootluse-pohimotted. [Kasutatud 22. märts 2021].

[58] Andmekaitse Inspektsioon, „Isikuandmed ja töötlemine,“ [Võrgumaterjal]. Available: https://www.aki.ee/et/eraelu-kaitse/isikuandmed-ja-tootlemine. [Kasutatud 22. märts 2021].

[59] Andmekaitse Inspektsioon, „Töötlemise õiguslikud alused,“ [Võrgumaterjal]. Available: https://www.aki.ee/et/eraelu-kaitse/tootlemise-oiguslikud-alused. [Kasutatud 22. märts 2021].

[60] Riigi Teataja, „Võlaõigusseadus,“ [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/104012021019. [Kasutatud 24. märts 2021].

[61] Andmekaitse Inspektsioon, „Andmete edastamine välisriiki,“ [Võrgumaterjal]. Available: https://www.aki.ee/et/teenused-poordumisvormid/andmete-edastamine-valisriiki. [Kasutatud 22. märts 2021].

[62] „Hoovi,“ [Võrgumaterjal]. Available: https://www.hoovi.ee/. [Kasutatud 20. veebruar 2021].

[63] RIK, „e-kinnistusraamat (XML teenus),“ [Võrgumaterjal]. Available: https://www.rik.ee/et/e-kinnistusraamat/xml-teenus. [Kasutatud 20. veebruar 2021].

[64] Hoovi, „Üürinõuete ost,“ [Võrgumaterjal]. Available: https://www.hoovi.ee/uurinouete-ost. [Kasutatud 9. mai 2021].

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

Mina, William Straus

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „REST-põhiste mikroteenuste väljatöötamine Amazoni veebiplatvormil krediidiriski hindamiseks“, mille juhendaja on Einar Kivisalu
   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.

14.05.2021

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.