TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Liisa Viljaste 174455IDSR

MIKROTEENUSTE DISAIN RIIGISEKTORIS EESTI HARIDUSE INFOSÜSTEEMI ÜLDHARIDUSE LÕPUDOKUMENTIDE RIIKLIKU REGISTRI NÄITEL

Diplomitöö

Juhendaja: Kristjan Karmo
Magistrikraad

Tallinn 2020

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: Liisa Viljaste

18.05.2020

Annotatsioon

Käesoleva diplomitöö põhieesmärgiks on uurida ja analüüsida mikroteenuste arhitektuuri disaini põhimõtteid ning nõudeid disaini tulemusel loodavale dokumentatsioonile ning leida erinevused nende rakendamisel riigisektoris. Mikroteenuste arhitektuuri disaini ja dokumenteerimise näitlikustamiseks riigisektoris käsitletakse tagasüsteemide analüüsi Eesti Hariduse Infosüsteemi üldhariduse lõpudokumentide riiklikus registris.

Mikroteenuste disaini teoreetilise käsitluse käigus uuritakse mikroteenuste arhitektuuri omadusi võrreldes teiste tarkvaraarenduse viisidega ning tuuakse välja mikroteenuste eelised ja puudused. Seejärel vaadeldakse mikroteenuste disainis olulisi põhimõtteid ning tuuakse välja dokumenteerimisel olulised aspektid ning soovituslik struktuur. Viimaks uuritakse disaini ja dokumenteerimise erisusi riigisektoris.

Üldhariduse lõpudokumentide teenuse disainil EHISes rakendatakse teoreetilise käsitluse tulemusi. Tuuakse näiteid rakendamise tulemustest andmete ning liideste kirjeldamisel ja vaadeldakse tekkinud probleeme. Järelduste osas tehakse kokkuvõte üldhariduse lõpudokumentide teenuse disaini tulemustest. Samuti soovitatakse tulevasi uurimisteemasid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 4 peatükki, 6 joonist, 3 tabelit.

Abstract

Microservice Design in the Public Sector, Based on the National Register for General Education Graduation Documents in the Estonian Education Information System

The aim of this work is to research and analyse principles of microservice design and requirements for documenting the results of this design. Also, the thesis tries to find distinctions in implementing these principles and requirements in the public sector.

Microservice design and specification in the public sector is approached with an example of back-end analysis of the National Register for General Education Graduation Documents in the Estonian Education Information System.

The thesis approaches the theory of microservice design by looking at the characteristics of microservice architecture compared with other software development styles (monolith and service-oriented architecture) and highlights its benefits and shortcomings. Secondly, the principles of microservice design are examined resulting with important aspects and recommended structure for the documentation. For example, using business context to define services and designing and documenting the interfaces based on user's needs. Finally, the thesis explores the features of microservice design and documentation in the public sector.

The results from this theoretical approach are implemented in the design of the service for General Education Graduation Documents. The thesis includes examples for data and interface specification and difficulties that rose during analysis. Detailed example of documenting the graduation document interface is looked at. The conclusion of this chapter gives an overview of the results of the design process. Finally, additional research areas in the future are suggested.

The thesis is in Estonian and contains 34 pages of text, 4 chapters, 6 figures, 3 tables.

Lühendite ja mõistete sõnastik

EHIS	Eesti Hariduse Infosüsteem on Haridus- ja Teadusministeeriumi poolt hallatav riiklik register hariduse andmete kogumiseks ja kättesaadavaks tegemiseks.
Haridusportaal	Hariduse, õppe- ja karjäärinõustamise infot koondav valdkonnaportaal.
Riigisektor	Hõlmab riigi valitsemise ja haldamisega tegelevad asutusi ja ettevõtteid ning nende funktsioone.
Spetsifikatsioon	Tehniline dokument, mis kirjeldab toote (loodava tarkvara) ehitust.
Tagasüsteem (ingl. keeles *back-end*)	Viitab süsteemi andmeid töötlevale, talletavale ning käitlevale osale, millele vastukaaluks võib välja tuua eessüsteemi (ingl. keeles *front-end*), mis tegeleb nende andmete esitlemisega.
Tarkvara arhitektuur	Tähistab tarkvara süsteemi põhistruktuuri ning tegeleb selle struktuuri loomisega.
Tarkvara disain	Tegeleb tarkvara süsteemide funktsionaalsuse kavandamisega.
Teenus	Mikroteenuste puhul komponent
Üldhariduse lõpudokument	Põhikooli või gümnaasiumi lõpetamist tõestav dokument, mis koosneb tunnistusest (põhidokument) ja hinnetelehest (lisadokument).

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Eesti Hariduse Infosüsteem (EHIS) on riiklik register, mis koondab haridussüsteemi puudutavaid andmeid. EHISe registrisse on üldhariduse lõpudokumentide andmeid kogutud alates 2004. aastast. 2019. aastal algas projekt EHISe üldhariduse lõpudokumentide lahenduse täiendamiseks ning paralleelselt alustati ka EHIS2e projektiga, mille eesmärgiks on EHISe süsteemis olemasoleva funktsionaalsuse järkjärguline üleviimine mikroteenustele. Üks esimesi uusi funktsionaalsusi, mis mikroteenuste arhitektuuri järgides luuakse on seotud üldhariduse lõpudokumentide registri täiendamise projektiga. Seoses selle projektiga tekkisid ka antud töös käsitletavad uurimisteemad.

Diplomitöö autor on alates 2018. aastast osalenud EHISe arendusprojektides analüütikuna ning osales üldhariduse lõpudokumentide lahendust täiendavas projektis tagasüsteemi (ingl. keeles *back-end*) analüütiku rollis, mis hõlmas endas nõuete kogumist, tagasüsteemi funktsionaalsuse analüüsi ja tehnilist kirjeldust lähtuvalt süsteemi kavandatud arhitektuurist. Arhitektuuri kavandamine ei kuulunud autori tööülesannete alla.

Töö sissejuhatuses antakse ülevaade EHISe üldhariduse lõpudokumentide registri täiendamise projekti taustast ja eesmärkidest. Lisaks tutvustatakse EHISe süsteemis mikroteenuste arhitektuuri kasutusele võtmise tagamaid ning planeeritud lahendust. Viimaseks sõnastatakse käesoleva töö eesmärk.

Teine peatükk hõlmab mikroteenuste disaini teoreetilist käsitlust. Selles peatükis peatutakse mikroteenuste arhitektuuri omadustel ning tuuakse välja selle eelised ning puudused. Lisaks vaadeldakse mikroteenuste disaini ning selle dokumenteerimise põhimõtteid. Peatüki lõpus tutvutakse mikroteenuste disaini eripäradega riigisektoris.

Kolmandas peatükis kirjeldatakse mikroteenuste disaini põhimõtete kasutamist EHISe üldhariduse lõpudokumentide teenuse kavandamisel. Peatükis tuuakse näiteid disaini tulemusel tekkinud andmete, liideste ja andmete töötlemise reeglite kirjelduste kohta.

Samuti tuuakse välja disaini käigus tekkinud probleemid ning pakutakse välja parendusi tulevikuks.

Lõpuks antakse kokkuvõttes ülevaade töö tulemustest.

1.1 Üldhariduse lõpudokumentide registri täiendamise projekt

Üldhariduse lõpudokumendid põhikooli ja gümnaasiumi lõpetajatele antakse täna välja paberkandjal. Õppeasutus sisestab dokumendile trükitavad andmed EHISesse ning trükib seejärel lõpudokumendi spetsiaalsele nummerdatud plangile. Lõpudokumentide omanikud kasutavad neid õpingute jätkamiseks või tööle kandideerimisel, samuti teatud töökohtadel kvalifikatsiooni omandamise tõendamiseks.

2018. aastal Trinidad Wiseman OÜ poolt teostatud ärianalüüs tõi välja järgmised kitsaskohad praeguses lõputunnistuste välja andmise protsessis ja kasutamises (Trinidad Wiseman OÜ 2018, lk 11-12):

- lõpudokumentide vormistamine ja trükkimine on ajamahukas ning seda on võimalik teostada ainult kindla ajaakna jooksul;
- paberdokumentide kadumisel või kahjustumisel on vajalik dokumendile duplikaat väljastada;
- dokumendile ligipääsuks on vajalik füüsiline ligipääs.

Üldhariduse lõpudokumentide lahenduse täiendamise projekti eesmärgiks on lahendada dokumentide säilitamise ja kasutamisega seotud kitsaskohti – täpsemalt peaks näiteks lõpudokumentide omanikele tekkima järgmised võimalused:

- tekib võimalus näha oma dokumentide andmeid läbi Haridusportaali;
- tekib võimalus teha registris olevate andmete kohta väljavõte;
- tekib võimalus jagada kolmandatele osapooltele ligipääsu dokumendi andmetele;
- lisaks saab registri andmete põhjal dokumentide andmetega tutvudes veenduda nende autentsuses ja kehtivuses.

Lahenduses kasutatakse juba EHISesse sisestatud andmeid, seega dokumentide väljaandjate töökoormus seoses uue lahendusega ei peaks tõusma.

Lisaks eelpool mainitud ärianalüüsile lähtuti lahenduse välja töötamisel EHIS2e arhitektuurist, kehtivatest seadustest ning EHIS1 süsteemis teostatud lahendusest.

Projekti tulemusel välja töötatav põhifunktsionaalsus on osa EHISe üldhariduse lõpudokumentide registri uue lahenduse pilootprojektist. Tulevastes etappides võiks teostada kutse- ja kõrghariduse lõpudokumentide ning kutsetunnistuste integreerimise samasse lahendusse, kuid antud projektis selle lahenduse detailidesse ei laskutud. Siiski arvestati võimalusel välja töötatava lahenduse puhul võimalike tulevaste vajadustega.

1.2 Mikroteenuste arhitektuuri kasutusele võtmine EHISe süsteemis

EHIS2 raamhanke tehnilise kirjelduse all (Haridus- ja Teadusministeerium 2018, lk 14) tutvustatakse süsteemi järgmiselt: „EHIS on riigi infosüsteemi kuuluv andmekogu, mille eesmärk on koguda informatsiooni haridussüsteemi korraldamiseks ning sihipärasemaks juhtimiseks. Registri andmeid kasutatakse (üli)õpilastele, õpetajatele ning õppejõududele teenuste ja toetuste pakkumiseks ning riikliku ja valdkondliku haridusstatistika koostamiseks ning uuringute läbiviimiseks.“

EHISe riiklik register võeti kasutusele 2004. aastal ning on siiani aktiivses kasutuses ning arenduses. Olemasolev süsteem on teostatud moodulipõhiselt ning kasutusel olevaid mooduleid on praeguse seisuga 14 (Haridus- ja Teadusministeerium 2018, lk 15-16).

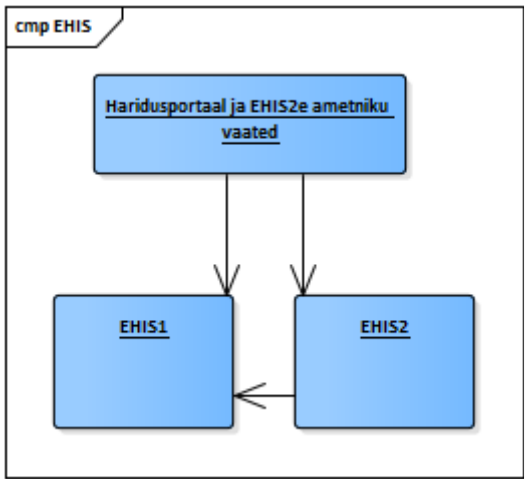
EHISe koosvõime tõstmiseks, vananenud tehnoloogiatest vabanemiseks ning jätkusuutlikkuse parandamiseks alustati 2019. aastal EHISe süsteemi viimist uuele tehnoloogilisele lahendusele, mille kavandamisel tuli järgida järgmiseid nõudeid arhitektuurile (Haridus- ja Teadusministeerium 2018, lk 16):

- loodav infosüsteemi uus osa peab olema teenusepõhine (infosüsteemiga andmevahetus peab olema võimalik ka ainult X-tee teenuste vahendusel, ilma kasutajaliideseta, et suurendada koosvõimet teiste infosüsteemidega ning vähendada dubleeriva andmesisestuse vajadust);

- kasutajaliides on osa Hariduportaali lahendusest;

- menetlusprotsesside toetamisel piirdub EHIS2 süsteemi funktsionaalsus andmete vahetusega menetluskeskkonnaga – menetluskeskkonda EHIS2 süsteemi ei looda;
- andmemudel tuleb luua optimeeritult ja vältida andmete dubleerimist;
- arenduse käigus loodav kood ning äriloogika peavad olema ühtsed ning taaskasutatavad;
- sarnast funktsiooni täitvad infosüsteemid tuleks liita üheks infosüsteemiks (näiteks EHIS ja kutseregister);
- olemasolevaid äriprotsesse tuleks ühtlustada ja optimeerida.

Selle tulemusel valitud lahendust iseloomustab mikroteenuste arhitektuur. Uuele tehnoloogilisele lahendusele üleminek on planeeritud selliselt, et olemasoleva süsteemi kõrvale luuakse uus süsteem – EHIS2, millesse hakatakse olemasolevat funktsionaalsust järkjärgult üle viima. Uus osa kasutab vajadusel vanas osas olevaid andmeid. EHIS2 süsteemi kasutav kasutajaliides integreeritakse Haridusportaali luues sinna kõrvale ka uued EHISe ametniku vaated. Planeeritud lahendust illustreerib järgnev joonis (Joonis 1).



Joonis 1 EHISe vana ja uus süsteem ning kasutajaliides.

Üks esimene komponent, mis EHISes otsustati teostada mikroteenusena, oli üldhariduse lõpudokumentide lahendus. Tegemist on täiesti uue funktsionaalsusega, mida vana süsteem ei paku. Diplomitöös käsitletakse mikroteenuste kavandamise ja kirjeldamisega seotud probleeme EHISes uurides mikroteenuste disaini ja spetsifitseerimise põhimõtteid ning tuues välja riikliku registri disainimise eripärad. Teemale lähenetakse tehnilise

spetsifikatsiooni loomise vaatepunktist ning ei peatuta pikemalt arhitektuuri kavandamisega seotud probleemidel.

Diplomitöö eesmärk on kirjeldada EHISe üldhariduse lõpudokumentide registri uut lahendust ning sellega luua näidis mikroteenuste kavandamisest ja kirjeldamisest riigisektoris EHISe süsteemi näitel. Lisaks sõnastatakse kavandamise põhimõtted ja tuuakse välja soovitused mikroteenuste kavandamisel spetsifikatsiooni kirjutamiseks eesmärgiga, et neid üldistusi oleks võimalik võtta aluseks tulevastes projektides disaini otsuseid tehes.

2 Mikroteenuste disaini teoreetiline käsitlus

Diplomitöö eesmärkideni jõudmiseks alustatakse esmalt mikroteenuste olemuse uurimisega, et teha kindlaks mikroteenuste üldised omadused ning eelised teiste tarkvaraarenduse viiside ees. Selle läbi jõutakse mikroteenuste disainis oluliste põhimõteteni ja täpsustatakse dokumenteerimisega seotud tavasid. Viimaks uuritakse, kas riigisektori disainipõhimõtted eristuvad üldistest põhimõtetest.

2.1 Mikroteenuste arhitektuur, selle eelised ja puudused

Mikroteenuste arhitektuuri on võetud kasutusele järjest laialdasemalt, teiste seas pakuvad oma teenuseid antud lähenemist kasutades ka sellised rahvusvahelised ettevõtted nagu Amazon, Netflix jt (Fulton 2015, Ismail 2018).

James Lewis ja Martin Fowler (Lewis ja Fowler 2014) iseloomustavad mikroteenuseid järgmiselt: „Kokkuvõtvalt on mikroteenused tarkvara arenduse viis, millega rakendust arendatakse väikeste teenuste kogumikuna, kus igal teenusel on oma protsess ning suhtlus toimub kasutades õhukesi protokolle, näiteks HTTP. Teenused on ehitatud ärilise võimekuse toetamiseks ning neid on võimalik üksikult paigaldada läbi automatiseeritud paigaldusmehhanismi. Teenuste tsentraalne juhtimine on viidud miinimumini. Teenused ise võivad olla kirjutatud erinevates programmeerimise keeltes ning kasutada andmeallikana erinevaid tehnoloogiaid.“

Selle peatüki käigus vaadeldakse erinevaid põhimõtteid tarkvara arhitektuuris, et saaks välja tuua mikroteenuste eelised ja puudused teiste tarkvaraarenduse viiside ees ning läbi nende eeliste ja puuduste jõuda mikroteenuste disainis oluliste aspektideni.

2.1.1 Tarkvara arhitektuuri evolutsioon

Antud töö raames ei ole eesmärgiks tutvustada detailselt kõiki tarkvara arhitektuuri evolutsiooni teid, kuid mikroteenuste omaduste paremaks illustreerimiseks peatutakse kahel teisel tarkvaraarhitektuuri lähenemisel: monoliitne arhitektuur ja teenusepõhine arhitektuur.

Monoliitses süsteemis moodustavad andmed ja neid kasutav äriloogikat sisaldav rakendus koos oma kasutajaliidesega ühe terviku. Sellest tulenevalt mõjuvad süsteemis tehtavad muudatused kogu süsteemile ning muudatuse tagajärjel võib avalduda vigu süsteemi osades, mis pealtnäha ei olnud muudatusega kuidagi seotud. See tähendab, et muudatuste tegemiseks on vaja omada head ülevaadet kogu süsteemist – see ei pruugi olla probleemiks arenduse algfaasis, kuid võib muutuda probleemiks kui arendajad vahetuvad ning süsteemi keerukus kasvab. Monoliit sõltub tugevalt ka valitud tehnoloogiatest – kui valitud tehnoloogiate toetamine lõpetatakse või muutub nende hinnastamine, siis võib muutuda süsteemi üleval pidamine väga kulukaks või tekkida vajadus alustada kogu süsteemi arendamist otsast peale. Lisaks, monoliidi skaleerimine tähendab üldjuhul kogu süsteemi seadistamist vastavalt tippkoormusele.

Monoliitse tarkvaraga kaasnevaid riske saab maandada selle modulaarseks muutmisega, kuid seoses ajas süsteemi lisanduva keerukusega võivad moodulite piirid hägustuda või tekkida vajadus suurematekstuudatusteks, et moodulite piirid uuel moel defineerida – algselt valitud moodulite puhul ei osatud ennustada tekkivaid vajadusi. Samuti on modulaarse monoliidi osi teistes süsteemides keeruline taaskasutada: takistavad süsteemides kasutatavad erinevad tehnoloogiad ja raamistikud; mooduli ülesehitusse võib imbuda valdkonna spetsiifika, mis takistab mooduli taaskasutamist teistes valdkondades.

Monoliitse rakenduse ehitamine on algfaasis tihti kiirem ning lihtsam, samuti sellega kaasnev haldus. Seega, monoliitse rakenduse ehitamine võib teatud juhtudel siiski õigustatud olla, näiteks kui ehitatav rakendus täidab kindlat lühiajalist eesmärki (Fowler, MonolithFirst 2015).

Avalikus sektoris need monoliidi eelised siiski enamasti ei avaldu, sest enamus riigi poolt pakutavatest teenustest on laiaulatuslikud ning seoses sellega tekib vajadus integratsiooniks teiste süsteemidega (Vaher 2020, lk 64).

Monoliitsele tarkvarale vastukaaluks hakkas 2000ndate alguses seoses veebipõhiste teenuste kasutusele võtmisega laiemalt levima teenusepõhine arhitektuur (ingl. keeles s*ervice-oriented architecture* ehk *SOA*). See on näide hajusast arhitektuurist, kus süsteemi osad on jagatud komponentideks, mille ehituses võivad olla kasutusel erinevad tehnoloogiad ja raamistikud. Komponendid teevad koostööd kasutades selgelt

määratletud liideseid, mida nimetatakse *API*ks (ingl. keeles *application programming interface*).

Teenusepõhise arhitektuuri puhul on süsteemi komponentideks teenuste pakkujad, teenuste tarbijad ja teenuste registrid. Teenust pakkuv komponent võib samaaegselt olla ka mõne teise teenuse tarbijaks. Sellises süsteemis on äriprotsessid teostatud teenusepõhiselt – teenuseks on teenuse pakkuja poolt pakutav ja tarbija jaoks vajalik äriline tegevus ning teenuste omavaheliste seoste ja töö tulemuse kaudu joonistuvad välja süsteemi poolt toetatud äriprotsessid.

Komponendi selgelt määratletud liidesed piiritlevad selle selgemalt eraldi tükiks, mille sisemine ehitus ei ole oluline sellega seotud komponentide jaoks. Sellise piiritlemise eeliseks on see, et komponendi sees tehtavate muudatuste mõju on samuti piiratud – vähemalt seni, kuni muudatuste tulemusena ei muutu komponendi liides. Teiseks on võimalik süsteemi skaleerida komponentide kaupa. Lisaks, kuna liidesed on defineeritud komponendist kasutatavatest tehnoloogiatest sõltumatult, siis on võimalik komponente taaskasutada ka erinevates süsteemides.

Kokkuvõtvalt võib öelda, et võrreldes monoliitse arhitektuuriga kasvab teenusepõhise arhitektuuriga süsteemi jätkusuutlikkus, kuid lisandub keerukust süsteemi ehitamisel ja haldamisel.

Eesti riigisektori tuntuim näide teenusepõhise arhitektuuri rakendamisest on ilmselt X-tee. X-tee kaudu on ühendatud erinevaid tehnoloogilisi lahendusi kasutavad riigiasutuste andmekogud. X-tee lahendus erineb klassikalisest teenusepõhisest arhitektuurist selle poolest, et igal X-teega ühendatud süsteemil peab olema turvaserver ja suhtlus kahe andmekogu vahel toimub mõlema süsteemi turvaserverite vahendusel (Vaher 2020, lk 72-73).

2.1.2 Mikroteenuste arhitektuur

Sarnaselt teenusepõhisele arhitektuurile on ka mikroteenuste arhitektuur hajus ning peegeldab seega hajusa arhitektuuri üldiseid omadusi. Mikroteenuste arhitektuuri puhul on hajusa süsteemi komponentideks ärivaldkonnapõhised teenused, mille suhtlus teiste komponentidega toimub *API*de kaudu. Kõige suurem erinevus teenusepõhise arhitektuuri ja mikroteenuste arhitektuuri vahel on viimase rõhk peeneteralisusele.

Oma raamatus „*Building Microservices*“ (Newman 2015, ptk 1) defineerib Sam Newman mikroteenuseid kui väikeseid, sõltumatuid teenuseid, mis teevad koostööd. Ta täpsustab, et teenused peavad olema nii väikesed kui vaja ja mitte väiksemad, sest mida väiksemad on teenused, seda enam võimenduvad mikroteenuste eelised, aga ka puudused. Sõltumatus tähendab aga, et mikroteenused on eraldi olemid ning teenuste vahelise seotuse vähendamiseks peab jälgima, mida teenused teevad liidese kaudu avalikuks ja mis peaks jääma peidetuks.

Newman soovitab mikroteenustest mõelda kui spetsiifilisest lähenemisest teenusepõhisele arhitektuurile, mis kasutab ära paranenud arusaamu süsteemidest ja arhitektuurist selleks, et teenusepõhist arhitektuuri hästi rakendada (Newman 2015, ptk 1).

Artiklis „*Microservices – a definition of this new architectural term*“ (Lewis ja Fowler 2014) tuuakse välja, et mikroteenuste arhitektuuril on kas kõik või vähemalt enamus järgmistest omadustest.

- Komponentideks on teenused.
- Teenused ja meeskonnad on organiseeritud ärivaldkonna järgi.
- Esikohal on toode, mitte projekt.
- Liidesed (ingl. keeles *endpoints*) on targad ja nö torud (sõnumite edastamise keskkonnad) on rumalad.
- Teenuste juhtimine on detsentraliseeritud.
- Andmete haldus on detsentraliseeritud.
- Infrastruktuur on automatiseeritud.
- Kavandamisel on arvestatud tõrgetega.
- Kavandamisel arvestatakse evolutsioonilise disaini põhimõtetega.

Kuna nii teenusepõhine arhitektuur kui ka mikroteenuste arhitektuur on hajusad, siis korduvad mikroteenuste puhul mitmed teenusepõhise arhitektuuri eelised, näiteks:

- võimalus kasutada süsteemis erinevaid tehnoloogiaid;
- võimalus skaleerida süsteemi komponendi kaupa;
- muudatuste mõju on piiritletud komponenti ning neid saab paigaldada teenuste kaupa;
- teenused on taaskasutatavad;
- süsteem on jätkusuutlikum.

Lisaks võiks laiendada mõningaid eelnevalt välja toodud eeliseid Newmani käsitluses mikroteenuste kontekstis järgmiselt (Newman 2015, ptk 1).

- Taaskasutatavusega seoses tuleb arvestada oluliselt mitmekesistunud teenuste kasutamise viiside ja eesmärkidega, näiteks tuleb mõelda kuidas samu teenuseid saaks kasutada lauaarvutis, mobiilis, tahvelarvutis jt nutiseadmetes.
- Jätkusuutlikkusega seoses on lisaeeliseks see, et komponentide väiksusest tulenevalt on selle välja vahetamine või käibelt eemaldamine vähem riskantne.
- Teenuste ja meeskonna organiseerimine ärivaldkondade järgi aitab hoida meeskonna suurust optimaalsena ning erinevate teenustega saavad paralleelselt töötada mitmed meeskonnad.

Mikroteenuste arhitektuuri puudusteks on sarnaselt teenusepõhisele arhitektuurile keerukuse kasv süsteemi ehitamisel ja haldamisel. Täpsemalt tuuakse artiklis „*Microservice Trade-Offs*“ (Fowler, Microservice Trade-Offs 2015) välja järgmised mikroteenuste puudused:

- hajusus – hajusaid süsteeme on keerukam programmeerida, sest kaugpäringud on aeglasemad ning alati on risk tõrgeteks;
- tulevikus saabuv kooskõla (ingl. keeles *eventual consistency*) – tugeva kooskõla säilitamine on hajusas süsteemis väga keeruline ja seetõttu tuleb hallata olukordi, kus süsteemis valitseb ajutine ebakõla;

- haldamise keerukus – paljude teenuste regulaarse (taas)paigaldamise haldamiseks on vajalik küps haldusmeeskond.

Need puudused võimenduvad vastavalt sellele, mida väiksematest teenustest süsteem koosneb, seega tuleb teenuste suuruse piiritlemisel hoolega mõelda meeskonna võimekusele selle keerukuse haldamisel.

2.2 Mikroteenuste disain

Mikroteenuste disaini uurides peatutakse antud töö raames peamiselt ärivaldkonna jagamisel süsteemi komponentideks ning nende komponentide vahelise suhtluse kavandamisel. Selleks toetutakse eelmises peatükis käsitletud mikroteenuste omadustele, eelistele ja puudustele. Süsteemi kavandamisega seotud tehnoloogilistel valikutel pikemalt ei peatuta, sest see ei täida antud töö eesmärke.

2.2.1 Disaini põhimõtted

Mikroteenuste disaini põhimõtted hõlmavad endas põhimõtteid komponentide ja nendevahelise suhtluse kavandamiseks. Komponentide piiritlemisel ärilise valdkonna kaudu on abiks valdkonnapõhise disaini (ingl. keeles *domain-driven design*) põhimõtted, mida tutvustatakse samanimelises raamatus (Evans 2003).

Valdkonnapõhise disaini lähenemise järgi ei peaks eraldama disainimudelit ja realisatsiooni mudelit. Selle asemel tuleks disainis, modelleerimisel, teostamisel ning igapäevases suhtluses kasutada ühtset keelt, mida mõistavad kõik projekti kaasatud huvigrupid – valdkonna ekspertidest arendajateni ning arenduse käigus tekkivad muudatused peaksid kajastuma ka selles ühises keeles. Koos ühise keele otsimisega otsitakse ka ärilise valdkonna sees paiknevaid ärilisi piire – piiritletud kontekste (ingl. keeles *bounded context*) (Evans 2003).

Piiritletud kontekstid ei ole niivõrd seotud andmetega kui pigem ärilise võimekusega – ühe piiritletud konteksti sisemiste protsesside ja reeglite tundmine ei ole oluline teise piiritletud konteksti raames. Piiritletud kontekstist saavad teenuse piirid. Selliselt grupeeritakse teenusesse kokku funktsionaalsus, mis võiks muutuda samaaegselt. Selle tulemusel on muudatussoovid funktsionaalsuses suurema tõenäosusega ühe teenuse piires ja ei mõjuta süsteemi laialdasemalt. Samuti võiks paigutada eraldi konteksti ajutise ärilise

funktsionaalsuse, näiteks lühiajalise kampaania või ühekordse üritusega seotud funktsionaalsuse.

Kui ühe piiritletud konteksti seest koorub välja mitu erinevat piiritletud konteksti, siis on võimalus sama konteksti sees eraldada mitu teenust, mis nende kasutaja vaates moodustavad siiski ühe terviku. Või tükeldadagi see komponent mitmeks väiksemaks. Selle otsustuse tegemisel on kasulik taaskord lähtuda reaalsest elust – kui välja koorunud teenuseid hakkaks haldama sama meeskond, siis on mõistlik need hoida ühises kontekstis (Newman 2015, ptk 3).

Piiritletud konteksti sisemiste protsesside ja reeglite tulemusena tekib siiski mingit avalikku infot, mida teised piiritletud kontekstid (teenused) vajavad oma tööks. Avalik osa tehakse teistele teenustele kättesaadavaks läbi liideste. Seejuures on oluline mitte avalikustada liiga palju teenuse sisemist ehitust ja loogikat, sest see tekitab suuremat seotust teenuse tarbijatega. Avaldama ei peaks ka näiteks mingi andmeolemi neid atribuute, mis puudutavad ainult teenuse sisemisi protsesse (Newman 2015). Liideste disainil tuleks pigem tuua paralleele reaalses äris toimuva andmevahetusega – teistele teenustele pakutakse andmeid nö raportite ja vormide näol, mis valmivad sisemiste protsesside tulemusel või on sisendiks neile sisemistele protsessidele. Liidesed peaksid olema tarbijale võimalikult lihtsalt kasutatavad ning teenus peaks pakkuma kõikidele tarbijatele vajalikku infot, mis puudutab selle teenuse konteksti.

Mikroteenuste arhitektuuri puhul on tüüpiliselt igal teenusel oma andmebaas (Lewis ja Fowler 2014). Sellisel moel on paremini täidetud põhimõte, et iga komponendi sisemus peaks jääma teiste komponentide eest peidetuks ning äriloogika, mida kasutatakse andmete õigeks haldamiseks, on kapseldatud ühte kohta. Samuti saab sellisel moel ülejäänud süsteemist sõltumatult vahetada välja teatud komponentide andmebaasi tehnoloogiaid.

Lisaks, võttes aluseks Lewise ja Fowleri poolt loetletud mikroteenustele omased jooned, millest oli juttu ka eelnevalt (lk 17), joonistuvad välja veel järgmised arhitektuuri aspektid, millele disaini ajal mõelda. Esiteks peaks teenuste vaheline suhtluskeskkond olema kavandatud võimalikult õhukeselt ning funktsionaalsus tuleb teenusteks tükeldada selliselt, et suhtlus teenuste vahel ei toimuks liialt tihedalt. Teiseks tuleb kavandamisel arvestada erinevate tõrkeolukordadega ning planeerida süsteemi käitumine erinevates

olukordades vastavalt sellele, kuidas soovitakse kasutaja kogemust nendes olukordades hallata. Samuti tuleks planeerida monitoorimiseks vajalikud mõõdikud (Lewis ja Fowler 2014).

Piiritletud kontekstide järgi teenuste piiritlemine on hea põhimõte, kuid algne disain võib olla keerukas. Sellepärast on räägitud ka strateegiast, kus uus funktsionaalsus teostatakse esmalt monoliidina ja jagatakse mikroteenusteks alles hiljem (Fowler, MonolithFirst 2015).

2.3 Disaini dokumentatsioon

Disaini tulemusel valmib kavand, mis enamasti tähendab mingil kujul olevat dokumentatsiooni. Lähtudes valdkonnapõhise disaini põhimõttest, et disainimudel ja realisatsioonimudel peaksid ühtima (Evans 2003, ptk 3), peaks dokumentatsioon peegeldama teostatavat lahendust. Sellest võiks lähtuda ka näiteks dokumentatsiooni struktuuri puhul ning struktureerida dokumentatsioon teenuste kaupa. Iga teenuse avalik osa (liidesed) peaks olema esitatud ühtlustatud moel üle kogu süsteemi dokumentatsiooni. Samuti tuleks süsteemi dokumentatsiooni hoida värskena, et ei tekiks lahknevusi disainimudeli ja realisatsioonimudeli vahel. Sellega seoses kasutatakse mikroteenustega koos tihti automaatlahendusi dokumentatsiooni genereerimiseks ja uuendamiseks. Näited sellistest automaatlahendustest on Swagger, API Blueprint jt.

Teenuse sisemiste funktsioonide ja protsesside kirjeldamine peab samuti vastama teostatavale lahendusele. See võib aga suuresti erineda vastavalt teenuse spetsiifikale. Seega, sisemise loogika kirjeldamisel võib küll lähtuda üldistest kokkulepetest, kuid ei peaks juurutama kohustuslikku standardit, mida kasutada üle kogu süsteemi dokumentatsiooni.

Teenuse kirjelduses joonistub välja seega kaks olulist osa:

- avaliku osa ehk liideste kirjeldus (sisaldab enamasti ka teenuse üldist kirjeldust),
- peidetud osa ehk teenusesiseste ärireeglite, andmete jms kirjeldus.

2.3.1 Liideste dokumentatsioon

Kui eelnevalt toodi välja, et liides tuleb disainida nii, et seda oleks tarbijal lihtne kasutada, siis sama põhimõtet tuleks kasutada ka liidese dokumentatsiooni puhul. See tähendab, et liidese kirjeldus peaks olema arendajale lihtsalt loetav, aga samas ka vormistatud nii, et see oleks üheselt mõistetav ning täielik. Liidese dokumentatsiooni kasutamise lihtsustamiseks on võetud kasutusele spetsifikatsiooni standardeid, näiteks OpenAPI, API Blueprint, RAML.

Olulisemad teenuse ja selle liideste dokumentatsiooni osad on:

- teenuse üldine kirjeldus;
- teenusega seotud liideste loetelu, kus meetodite abil joonistuvad välja võimalikud tegevused;
- iga liidese kohta
    - liidese üldine kirjeldus,
    - sisendparameetrid ja
    - väljundi skeem.

Lisaks eelnevatele võib dokumentatsioon sisaldada ka päringu sisendi ja väljundi näidiseid. Näidiste puhul võiks küll kasutada võimalikult palju näidisandmeid, kuid hoida need siiski realistlikud ning vajadusel tuua välja mitu näidet. Väljundi skeem või mudel peaks välja tooma iga elemendi kohta selle kirjelduse ja andmetüübi ning näitama ka andmete struktuuri. Lisaks võib dokumentatsioon sisaldada ka lühijuhendit, õpetusi ning interaktiivseid lahendusi päringute näitlikustamiseks (Johnson 2019).

2.3.2 Teenuse sisemine spetsifikatsioon

Teenuse sisemise loogika kavandamisele ja kirjeldamisele võiks autori arvates läheneda sarnaselt üldistele põhimõtetele tarkvara funktsionaalsuse kavandamisel ja spetsifitseerimisel ning kasutada tehnikaid, mis sobivad antud ärilise valdkonnaga. See tähendab, et spetsifikatsioon ei sõltu süsteemis kasutatavast tehnoloogilisest lahendusest, vaid kirjeldab mustreid ja reegleid süsteemi või andmete kasutamisel, mis antud ärilisele

valdkonnale omased. Paindlikkus annab võimaluse katsetada erinevaid analüüsitehnikaid, mis antud ärilises valdkonnas paremini võiksid sobida.

Kavandamisele ja spetsifitseerimisele lähenetakse tihti läbi nõuete kogumise, milleks on välja pakutud palju erinevaid tehnikaid, näiteks dokumentide analüüs, fookusgrupilt tagasiside kogumine, liidese analüüs, intervjuud, kasutajate jälgimine, töötoa korraldamine, olemasoleva lahenduse kasutajaliidese uurimine ja küsitlused (Wiegers ja Beatty 2013, ptk 7). Enamus juhtudel tuleb valdkonna nõuetest mitmekesisema arusaama saamiseks kasutada nõuete kogumiseks paralleelselt mitut tehnikat.

Kavandatava funktsionaaluses abil tehtavate tegevuste kirjeldamiseks kasutatakse näiteks kasutuslugusid (ingl. keeles *user story*), kasutusmalle (ingl. keeles *use case*) ja stsenaariumeid (ingl. keeles *scenarios*). Nende tehnikate kasutamise valik sõltub eelkõige sellest kellele ja milleks kirjeldus luuakse: kas projektijuhile tööde planeerimiseks; arendajale või testijale lahenduse teostamiseks ja testimiseks; disainerile lahenduse elulisemaks muutmiseks (Iaboni 2013). Tegevuste kirjeldused kuuluvad sisemise spetsifikatsiooni alla siiski ainult siis, kui nad kirjeldavad teenuse sisest käitumist. Kui tegevuste kirjeldusi kasutatakse teenuse avalikus vaates võimalike tegevuste illustreerimiseks, siis kuuluvad need liideste dokumentatsiooni alla.

Andmete ja nende vaheliste seoste täpsustamiseks kasutatakse põhiliselt järgmiseid tehnikaid: *ERD* (ingl. keeles *entity relationship diagram*), *UML* (ingl. keeles *unified modeling language*) ja andmesõnastik. Esimesed kaks on visuaalsed ning nende puhul kujutatakse andmeid diagrammina vastavalt kokku lepitud notatsioonile. Andmesõnastiku puhul aga kirjeldatakse andmeid tabeli abil, kus põhielementidena on välja toodud andmekogumite (tabelite) nimekirjad ning iga andmekogumi atribuutide (tabeli veergude) nimekirjad koos vastavate andmetüüpidega. Lisaks võib soovi korral lisada elementide kirjeldused, tabelite ja veergude vahelised seosed ning piirangud (näiteks unikaalsus, vaikeväärtused, võimalikud väärtused, arvutuslikud veerud) (Kononow 2019).

Siiski võiks jälgida teenuse sisemise loogika kavandamise ja kirjeldamise käigus, et konteksti sees kasutataks ühtset keelt (Evans 2003, ptk 2). See tähendab, et samu mõisteid kasutatakse samas tähenduses nõuete, ärireeglite, kasutusmallide, andmemudeli jne täpsustamisel ning kirjeldamisel.

2.4 Mikroteenuste disain riigisektoris

Eelnevalt peatuti mikroteenuste disaini ja spetsifikatsiooniga seotud üldistel põhimõtetel. Järgmises sektsioonis uuritakse kas need üldised põhimõtted vajavad riigisektoris kohandamist. Selleks peatutakse mikroteenuste riigisektori süsteemis kasutamise põhjustel, uuritakse kas mikroteenuste disainipõhimõtted sobivad riigisektorisse ning lõpuks tuuakse välja riikliku registri disainimise eripärad.

Riigisektori andmekogude ja infosüsteemide ning nende poolt osutatavatele teenuste kogumile viidatakse dokumendis „Eesti infoühiskonna arengukava 2020“ (Vabariigi Valitsus 2018, lk 25-27) kui riigi infosüsteemile. Riigi infosüsteemi käsitletakse tervikuna, et paremini juhtida üksikute infosüsteemide arengut ning sellega toetada e-riigi arengut. Riigi infosüsteemi arengu tegevussuundadeks on eelnevalt viidatud arengukava järgi kesksete komponentide ja koosvõime lahenduste arendamine, riigi infosüsteemi koosvõimelisena toimimise tugevdamine, andmete ja tehnoloogiate ühis- ja taaskasutatavuse edendamine, e-teenuste piiriülese osutamise ja kasutamise arendamine ning infoühiskonna arengute jälgimine. Täpsemalt on ühis- ja taaskasutatavuse suurendamiseks plaan võtta suund riigi infosüsteemis mikroteenuste põhise arhitektuuri juurutamiseks ning edendada *API*-liideste kaudu süsteemide ja andmete avamist ning ühiskasutust.

Kuigi arengukavas on mikroteenuste arhitektuur peamiselt välja toodud ühis- ja taaskasutatavuse eeliste tõttu, siis riigisektoriks mikroteenuste kasutamise eelisteks on ka muudatuste mõju vähendamine süsteemile ning jätkusuutlikkuse kasv, sest süsteemid on üldiselt keerukad ning neid kasutatakse pikaajaliselt. Samuti toob teenuste ühine kasutamine välja vajaduse süsteemi teenuse kaupa skaleerida.

Riigi infosüsteemide arendamisel puudub pikaajaline kogemus mikroteenustega ja selle tõttu on arvestatavaks riskiks nendes süsteemides mikroteenuste arhitektuuri kasutusele võtmisega seoses vähene suutlikus tulla toime mikroteenuste ehitamise ja haldamise keerukusega.

Kuna mikroteenuste arhitektuuri poolt pakutavad eelised ja puudused on ühtlaselt olulised ka riigi infosüsteemides, siis sellest lähtuvalt tuleks riigisektoris mikroteenuste disainil üldiselt lähtuda samadest põhimõtetest, mis iseloomustavad head mikroteenuste disaini. See tähendab, et ka riigisektoris on oluline teenuste piiritlemine ärilise konteksti järgi,

kuigi äri tähendab siin riigiasutuste poolt kodanikele, ettevõtetele, riigiametnikele ning teistele riigiasutustele pakutavaid teenuseid.

Seoses liideste disainiga toodi mikroteenuste disaini põhimõtete all välja vajadus teha liidesed teenuse tarbijatele kergelt kasutatavaks ning see vajadus on riiklike infosüsteemide puhul kindlasti samuti oluline, eriti kuna teenustel võib olla palju ning erinevaid kasutajaid. Osa sellest kasutamise lihtsusest võiks tulla riigisektoris ühtsete ja avatud standardite kasutusele võtuga API dokumenteerimisel, mille eeliseks on tihti ka võimalus kasutada API automatiseeritud dokumenteerimise lahendusi. Selline avatud standarditel põhinev lähenemine dokumenteerimisel on oluline just APIde osas, infosüsteemi sisemise ehituse disaini ja dokumenteerimise võib lahendada infosüsteemi põhiselt vastavalt ärivaldkonna spetsiifikale.

Eelnevate disaini aspektidega sobitub ka Eesti riigisektori süsteemide arendamisel võetud suund API-põhise arhitektuuri (ingl. keeles *API-first development*) poole (Kütt 2016), mille puhul esmalt luuakse API pidades silmas selle tulevase kasutaja huve (tulemuseks peab olema terviklik, hästi juhitav ning dokumenteeritud API) ning seejärel luuakse APIt kasutavad lahendused, näiteks veebileht või mobiilirakendus.

Mikroteenustega seotud riskide maandamiseks tuleks jälgida, et valdkondade teenusteks tükeldamisel ei mindaks esialgu liialt peeneteraliseks.

3 Üldhariduse lõpudokumentide teenuse disain

Käesolevas peatükis kirjeldatakse eelnevalt välja toodud mikroteenuste disaini ja spetsifikatsiooni loomise teoreetiliste põhimõtete rakendamist EHISe üldhariduse lõpudokumentide lahendust täiendavas projektis.

Üldhariduse lõpudokumentide lahenduse teostamine ühe esimese komponendina EHIS2 süsteemis on hea võimalus mikroteenuste arhitektuuriga tutvumiseks. Lisanduva funktsionaalsuse keerukus ei ole väga suur ning see on ärilise konteksti järgi kergelt kapseldatav. Seetõttu on selle komponendi kavandamise kaudu võimalik läbi proovida uute mikroteenuste kavandamise metoodikaid ning selle kapseldamine iseseisvasse *certificates* teenusesse ei nõua suurt vaagimist.

Teenuse komponendi sisemist mudelit uurides joonistub välja tinglikult eraldiseisva komponendina lõpudokumentidele jagatavate ligipääsude komponent. Ligipääsud vajavad siiski tähenduse omamiseks alati ka seotud lõpudokumendi andmeid. Seega ei ole otstarbekas ligipääse eraldi teenusena kavandada ega teenust edasi tükeldada, sest seosed ligipääsude ja lõpudokumentide teenuste vahel oleks nende eraldi kapseldamisel liialt tihedad.

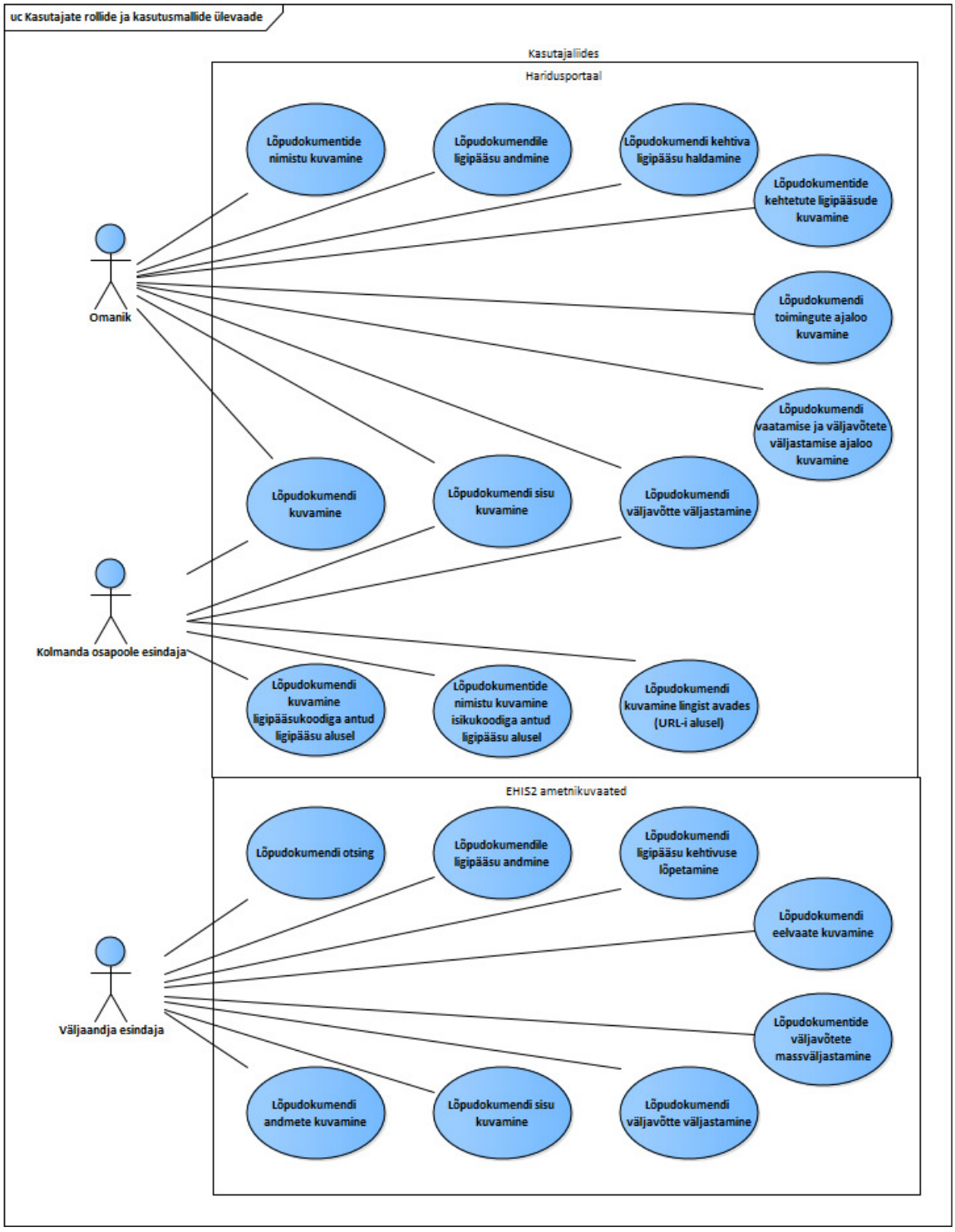
EHIS2 süsteemi dokumentatsioon on organiseeritud teenuste kaupa, et peegeldada teostatavat lahendust ning toetada tööde ja osalevate töötajate organiseerumist teenuse kaupa, sest ka meeskondade organiseerumine teenusepõhiselt kuulub mikroteenuste arhitektuuri põhimõtete alla. Seega on *certificates* teenuse kavandamisel loodud dokumentatsioon eraldiseisvas alamosas. Teenuse dokumentatsioonis on kirjeldatud teenuse liidesed ning nendega seotud ärireeglid ja teenuse andmete kirjeldus.

Vastavalt valdkonnapõhise disaini põhimõtetele värskendatakse teenuse kavandit kui analüüsi käigus mõisted muutuvad või ilmnevad uued asjaolud. Vajaduse tõttu kavandi dokumentatsiooni pidevalt värskena hoida kasutatakse EHIS2 süsteemi disaini käigus

loodava spetsifikatsiooni haldamiseks wiki-lahendust, millesse on sisse ehitatud versioonihaldus ning mis on kättesaadav kõigile projektis kaasalöövatele osapooltele.

Projekti skoobis olid nii tagasüsteemi funktsionaalsuse kui ka eessüsteemi loomine. Loodavast lahendusest parema tervikpildi saamiseks kirjeldatakse esmalt erinevaid kasutajate rolle ning nende süsteemi kasutamise malle eessüsteemi vaates. Seejärel liigutakse edasi autori töö tulemusel analüüsitud ja kirjeldatud tagasüsteemi funktsionaalsuse juurde.

Loodava funktsionaalsuse puhul on kolm olulisemat kasutajate rolli: lõpudokumentide omanikud, väljaandjate esindajad ning kolmandate osapoolte esindajad. Lõpudokumentide omanikel avaneb uues lahenduses võimalus enda dokumentide andmetega Haridusportaali kasutajaliidese kaudu tutvuda. Lisaks saavad nad võtta registri väljavõtte, et seda kolmandatele osapooltele jagada või anda kolmandatele osapooltele ligipääsu dokumendiga tutvumiseks Haridusportaalis. Väljaandjate esindajad saavad EHIS2e ametniku vaadete kaudu otsida väljastatud dokumente, vaadata dokumentide andmeid, võtta registriväljavõtteid ning anda dokumendile ligipääsu ligipääsukoodi alusel. Kolmandad osapooled saavad neile jagatud ligipääsu alusel tutvuda dokumentidega, millele neil on kehtiv ligipääs. Kasutajate rollide ja kasutusmallide ülevaade on nähtav järgneval joonisel (Joonis 2).



Joonis 2 Kasutajate rollide ja kasutusmallide ülevaade.

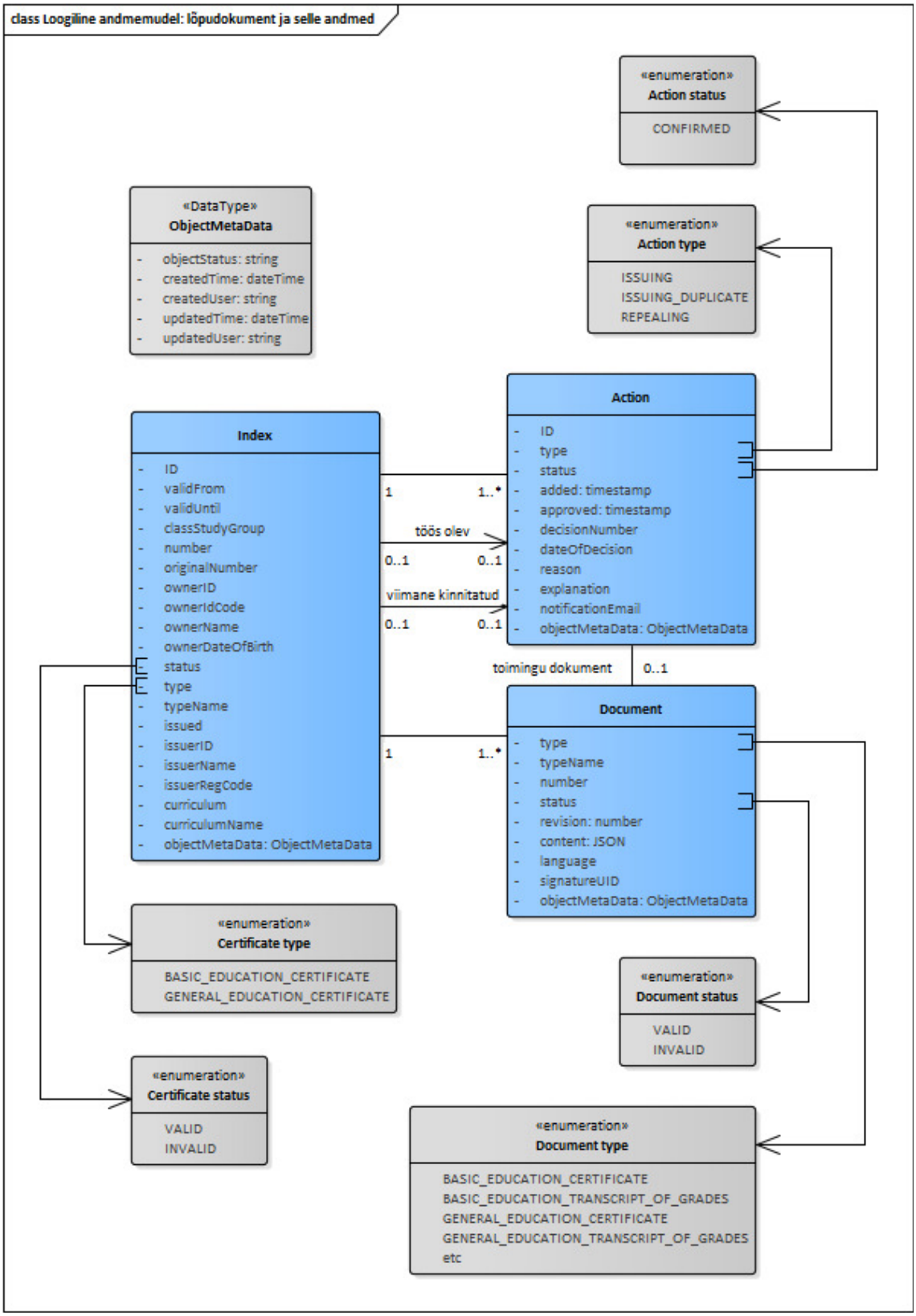
3.1 Andmed

Teenusega seotud andmed on kavandatud eraldiseisva andmebaasina ning andmete salvestamine, muutmine ning pärimine toimub ainult läbi teenuse poolt pakutavate liideste.

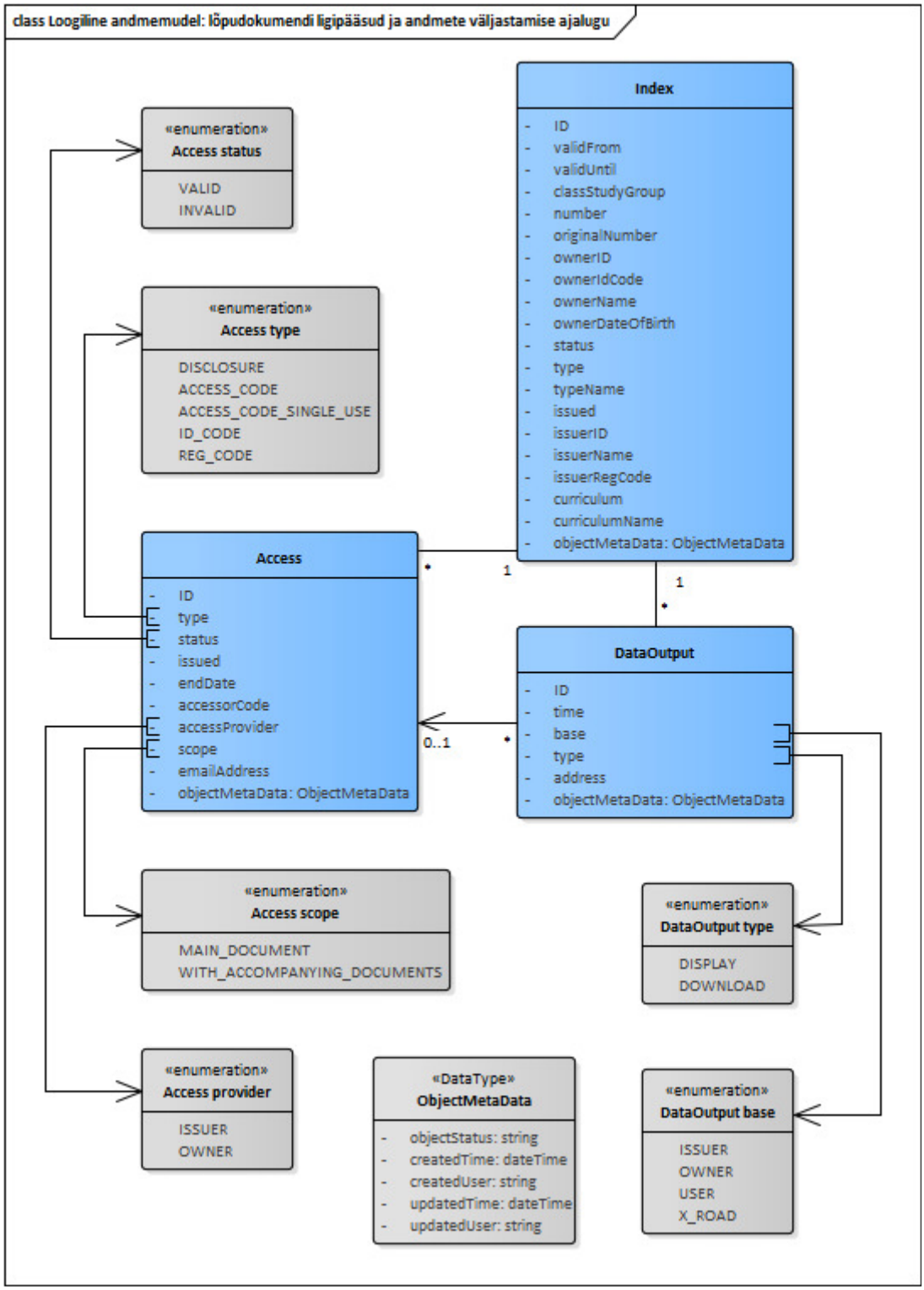
Andmed on modelleeritud *UML* notatsioonis loogilise andmemudelina. Lisaks on kasutatud andmesõnastiku tehnikat. See tähendab, et iga andmeolemi kohta on lisatud täpsem kirjeldus ka eraldiseisva wiki lehena, kus täpsustatakse andmeolemi atribuute, andmete tüüpi, kohustuslikkust, võimalikke ning vaikeväärtuseid jne. Samuti märgitakse iga andmeolemi ning atribuudi kohta lühike kirjeldus. Näide Document andmeolemi täpsema kirjelduse kohta on leitav Lisa 1 alt.

Teenuse andmemudeli parema loetavuse huvides on see jaotatud kahele joonisele, mis kujutavad järgmiseid andmeolemeid ja nende seoseid:

- lõpudokument, sellega seotud dokumentide ja tegevuste andmed (Joonis 3);
- lõpudokument, sellele antud ligipääsud ja andmete väljastamise ajalugu (Joonis 4).



Joonis 3 Lõpudokument, sellega seotud dokumentide ja tegevuste andmed.



Joonis 4 Lõpudokument, sellele antud ligipääsud ja andmete väljastamise ajalugu.

Liideste kaudu tehakse neist andmetest avalikuks selline komplekt, mis teenuse tarbijatele vajalik. Näiteks ei ole liideste kaudu kättesaadavad andmebaasi kirjete metaandmed (andmebaasi kirje lisamise ja muutmise aeg jms) ning kustutatud staatuses kirjed, kuna nende andmete välja näitamiseks puudub antud juhul äriline põhjus.

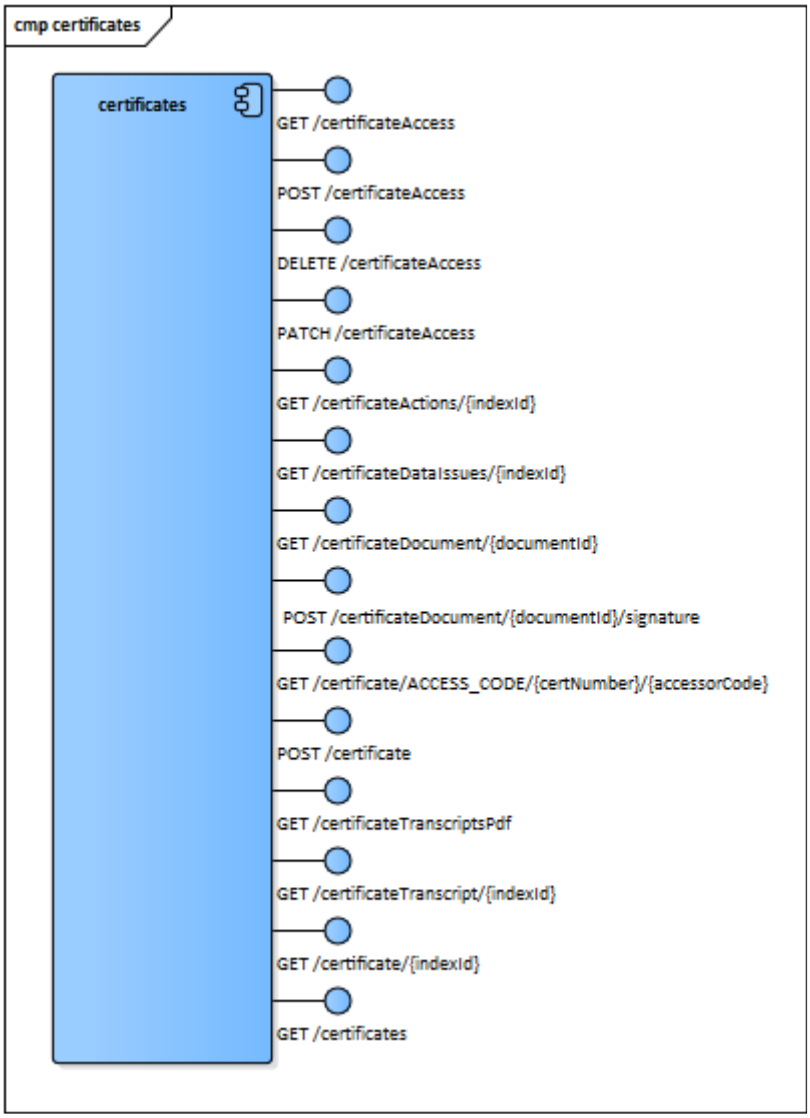
Teiselt poolt on liidesed kavandatud selliselt, et kasutajad saaksid kõik vajalikud andmed, mis kuuluvad teenuse poolt hallatavasse valdkonda, liideste kaudu kätte ning neid andmeid ei peaks olema vajadust veel omakorda töödelda. Näiteks on liidestes olemas kõik vajalikud filtrid tagastatavate andmete piiramiseks.

Küsimusi tekitas andmekirjete identifitseerimine ning identifikaatorite välja andmine teenuste kaudu. Hea praktika oleks andmebaasis kirjete identifitseerimiseks kasutatava primaarvõtme (ingl. keeles *primary key*) asemel kasutada andmete identifitseerimiseks ja pärimiseks ärilist identifikaatorit ning jätta primaarvõti ainult teenuse sisemistele protsessidele kasutamiseks. Lõpudokumentide puhul aga näiteks dokumendi number üksinda vajalikku unikaalsust ei paku, sest nummerdamise süsteemid on aastatega muutunud ning unikaalsust saaks seega luua ainult dokumendi numbri kombineerimisega teiste äriliste identifikaatoritega. Lõpudokumentide puhul oleks unikaalsuse saavutamiseks olnud vaja kombineerida nelja ärilist andmeelementi ning sellise keerukuse tekitamine ei olnud otstarbekas. Seega otsustati praeguses lahenduses kasutada andmebaasis genereeritavat järjekorra numbrit. Tulevikus võiks kaaluda sellisel puhul ka universaalselt unikaalse identifikaatori (ingl. keeles *universally unique identifier* e *UUID*) genereerimist, sest see peidab mõnevõrra paremini teenuse sisemist loogikat.

3.2 Liidesed

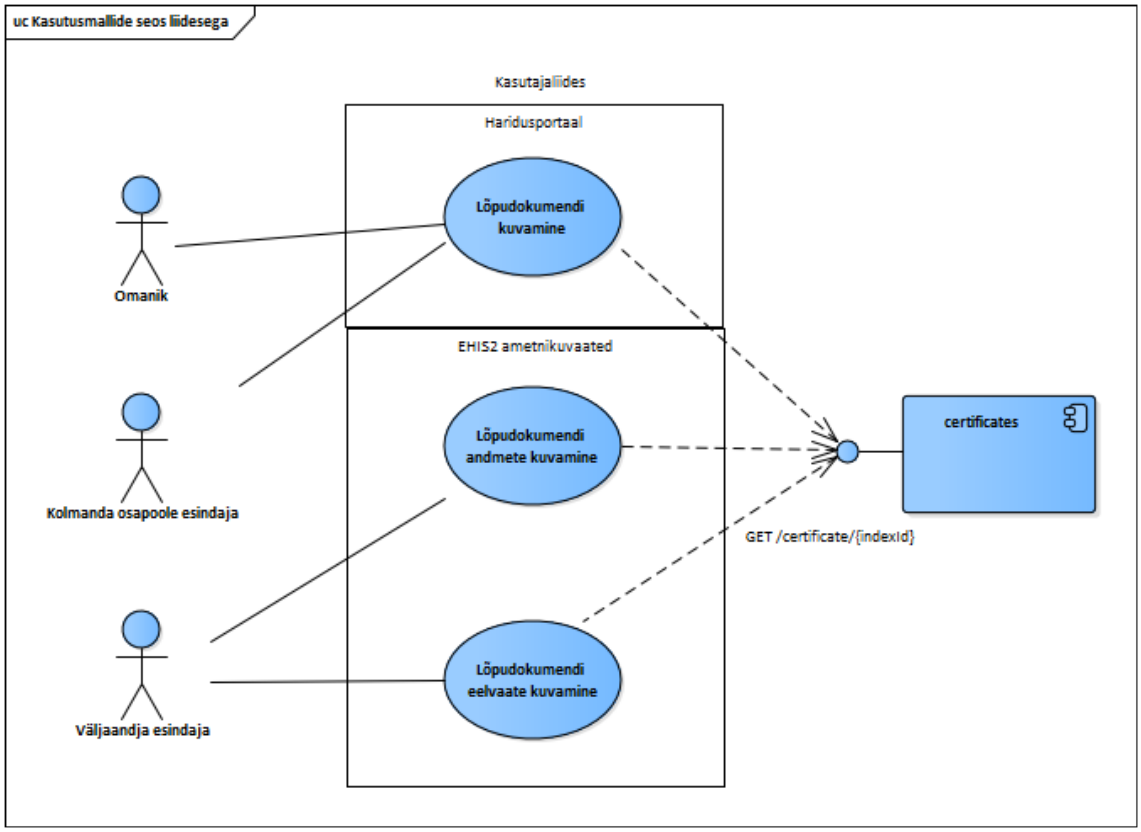
Teenuse liideste kirjeldamisele on lähenetud võimalikult ühtlustatult - iga liidese kirjelduses tuuakse välja liidese lühikirjeldus, kus on välja toodud ka mõned võimalikud kasutamise juhud. Samuti loetletakse liidese kirjelduses võimalikud sisendparameetrid, nende kirjeldus, mitmesus, võimalikud ning vaikeväärtused. Ka väljundi skeem on iga liidese spetsifikatsioonis kirjeldatud ning koosneb samuti andmeelementidest, nende struktuurist, mitmesusest, andmetüüpidest ning täpsustavatest selgitustest.

Teenusega suhtlemiseks ja funktsionaalsusega seotud toimingute tegemiseks defineeritud liidesed on nähtavad järgneval joonisel (Joonis 5).



Joonis 5 *Certificates* teenus ja selle liidesed.

Liidese üldkirjelduses on viide kasutusmalli(de)le, mille puhul antud liidest saaks kasutada teenusega suhtlemiseks ning rollidele, kes on nende kasutusmallidega seotud. Näiteks lõpudokumendi päringu liidese *GET /certificate/{indexId}* üldkirjelduses viidatakse kasutusmallidele „Lõpudokumendi kuvamine“, „Lõpudokumendi andmete kuvamine“ ja „Lõpudokumendi eelvaate kuvamine“. Need seosed kasutusmallide, rollide ja liidese vahel on nähtavad joonisel (Joonis 6).



Joonis 6 Kasutusmallide, kasutajate rollide ja liidese vahelised seosed.

Lisaks on oluliseks osaks teenuse liidese dokumentatsioonis võimalike sisendite ja väljundite kirjeldus. Sisendite puhul loetletakse liidese poole pöördumisel kaasa antavad andmed ning iga andmeelemendi puhul tuuakse välja kas tegemist on kohustusliku andmeelemendiga, näidatakse mitmesust, kirjeldatakse klassifikaatorite puhul võimalikud väärtused ning lisatakse vajadusel täiendavad selgitused andmete ärilise ja tehnilise tähenduse täpsustamiseks. Eelpool väljatoodud *GET /certificate/{indexId}* liidese sisendite kirjeldamise kohta on toodud näide järgnevas tabelis.

Tabel 1 Näide liidese *GET /certificate/{indexId}* sisenditest

Andmeelemendi nimi	**Mitmesus**	**Vaikeväärtus, valideerimine**	**Selgitus, täiendav tehniline info**
indexId	1		Lõpudokumendi identifikaator, mille detailandmeid päritakse.

accessType	0..1	Võimalikud väärtused: • „ID_CODE“	Ligipääsu tüüp, mis antakse sisendisse, juhul kui andmeid päritakse ligipääsu alusel.

Liidese väljundi kirjeldus loetleb sarnaselt sisendile pöördumise vastuses võimalikud andmed ning iga andmeelemendi puhul kirjeldatakse kas tegemist on kohustusliku andmeelemendiga, näidatakse mitmesust, klassifikaatorite puhul võimalikud väärtused ning lisatakse vajadusel täiendavad selgitused andmete ärilise ja tehnilise tähenduse täpsustamiseks. Liidese *GET /certificate/{indexId}* väljundi kirjeldamise kohta on toodud näide järgnevas tabelis.

Tabel 2 Näide liidese *GET /certificate/{indexId}* väljundist.

Andmeelemendi nimi	**Mitmesus**	**Selgitus, täiendav tehniline info**
role [1] – Päringu teinud kasutaja roll andmete pärimisel.		
role.**base**	1	Päringu alus, näidatakse kas andmeid päriti väljaandja, omaniku või kolmanda osapoolena. Võimalikud väärtused: • „ISSUER“ • „OWNER“ • „USER“
role.**accessScope**	0..1	Ligipääsu ulatus. Tagastatakse kui lõpudokumendi andmeid päritakse ligipääsu põhjal näitamaks kas ligipääsu põhjal on andmete kasutajal õigus näha ainult põhidokumendi andmeid või ka lisadokumentide andmeid. Võimalikud väärtused: • „MAIN_DOCUMENT“ • „WITH_ACCOMPANYING_DOCUMENTS“
index [0..1] – Leitud lõpudokumendi üldandmed.		
index.**id**	1	Lõpudokumendi identifikaator.
index. **classStudyGroup**	0..1	Klassi või õpperühma tunnus.

index.**number**	1	Lõpudokumendi kehtival redaktsioonil kuvatav number.
index. **originalNumber**	1	Algupärase lõpudokumendi number. Erineb kehtivast numbrist kui on väljastatud duplikaat.
index.**ownerIdCode**	0..1	Omaniku isikukood.
index.**ownerName**	1	Omaniku nimi.
index. **ownerDateOfBirth**	0..1	Omaniku sünnikuupäev.
index.**status**	1	Näitab kas lõpudokument on kehtiv või kehtetu. Võimalikud väärtused: • „VALID“ • „INVALID“
index.**type**	1	Näitab lõpudokumendi tüübi koodi. Võimalikud väärtused: • „BASIC_EDUCATION_CERTIFICATE“ • „GENERAL_EDUCATION_CERTIFICATE“
index.**typeName**	1	Näitab lõpudokumendi tüübi nimetust. Võimalikud väärtused: • „põhikooli lõputunnistus“ • „gümnaasiumi lõputunnistus“
index.**issued**	1	Välja andmise kuupäev, sisuliselt lõpudokumendi välja andmiseks tehtud otsuse kuupäev.
index.**issuerName**	1	Väljaandja õppeasutuse nimi.
index.**issuerRegCode**	1	Väljaandja õppeasutuse registrikood.
index.**curriculum**	1	Lõpetatava õppekava kood.
index. **curriculumName**	1	Lõpetatava õppekava nimetus.
index.**documents** [0..n] – Leitud lõpudokumendiga seotud dokumentide loetelu (metaandmed ilma JSON formaadis sisuta), näiteks tunnistus ja hinneteleht.		
index.documents.**id**	1	Dokumendi identifikaator.
index.documents.**type**	1	Dokumendi liigi kood.
index.documents. **typeName**	1	Dokumendi liigi nimetus.

index.documents. **number**	1	Dokumendi number.
index.documents. **status**	1	Näitab kas dokumendi redaktsioon on kehtiv või kehtetu. Võimalikud väärtused: • „VALID“ • „INVALID“
index.documents. **revision**	1	Dokumendi versiooni näitav number.
index.documents. **language**	1	Dokumendi keel.

Liidesega *GET /certificate/{indexId}* sarnasel moel on kirjeldatud ka kõik ülejäänud certifcates teenuse liidesed, kuigi olenevalt liidese kaudu tehtavast tegevusest kirjelduse keerukus mõnevõrra erineb. Näiteks lõpudokumendi andmete salvestamise liidese *POST /certificate* kirjeldus hõlmab kogu lõpudokumendi andmete ja nende valideerimise reeglistikku ning on see tõttu oluliselt keerukam; samas kui ligipääsu kehtetuks muutmise liides *DELETE /certificateAccess* on oluliselt lihtsam.

Kuna teenuse liideste wiki dokumentatsioon ei paku siiski alati teenuse kasutajale piisavalt formaalset ja standardset kirjeldust ning kiire tempo puhul on raske seda igal ajahetkel värskena hoida, siis paralleelselt wiki dokumentatsioonile võeti projektis kasutusele liideste automaatse dokumentatsiooni genereerimise lahendus Swagger. Nii pakub wiki leht teenuse kasutajale inimloetavat ja üldisemat kirjeldust ning automaatselt genereeritud dokumentatsioon annab formaalsema kirjelduse koos sisendite ja väljundite näidistega. Swaggeri kaudu genereeritud dokumentatsioon suurendab ka dokumentatsioonist ühtset arusaamist, sest vastab OpenAPI liideste dokumenteerimise avatud standardile.

3.3 Andmete töötlemise reeglid

Teenuse funktsionaalsuse täpsustamiseks valitud meetodid lähtuvalt ärilisest valdkonnast ja eesmärgile sobivusest. Antud juhul koguti nõudeid kasutades dokumentide analüüsi, struktureerimata intervjuusid ning olemasoleva lahenduse uurimist EHISe süsteemis. Näiteks lõpudokumentide andmekomplektide ja nendega seotud reeglite kindlaks tegemiseks kasutati Vabariigi Valitsuse määrust „Põhikooli ja gümnaasiumi

lõputunnistuse ning riigieksamitunnistuse statuut ja vormid“ (Riigi Teataja 2018) ning uuriti EHIS1 süsteemi salvestatavaid andmeid. Uurimise käigus tekkinud küsimustele vastuste saamiseks intervjueeriti valdkonna eksperte.

Kogutud nõuete põhjal loodi teenuse sisemise loogika kirjeldamiseks kasutusmallid, mis kirjeldavad teksti kujul erinevaid stsenaariumeid teenuse poole pöördumisel. Kuna teenuse poole pöördumine toimub läbi liideste, siis on ka kasutusmallid kirjeldatud liideste kaupa. Jätkates liideste kirjeldamise all toodud lõpudokumendi päringu liidese *GET /certificate/{indexId}* kirjeldust, siis selle liidese kasutusmalli kirjeldus on nähtav järgmises näites (Tabel 3), kus on näha sammude kaupa kirjeldatud tegevused (näites rasvases kirjas) ning nende tegevuste tehnilisi täpsustusi.

Tabel 3 Liidese *GET /certificate/{indexId}* kaudu tehtavate päringute töötlemise kirjeldus.

Samm	**Tegevus**
1.	**Teenus otsib päritava lõpudokumendi selle identifikaatori järgi.** Sisendis antakse indexId.
2.	**Teenus tuvastab kasutajal õiguse lõpudokumendi andmeid pärida lõpudokumendi omanikuna.** Lõpudokumendi ownerIdCode vastab sisse loginud kasutaja isikukoodile, kasutaja rolliks väljundisse määratakse role.base = "OWNER".
a.	**Teenus tuvastab kasutajal õiguse lõpudokumendi andmeid pärida väljaandja esindajana.** Lõpudokumendi väljaandja (issuerID) vastab sisse loginud kasutaja poolt esindatavale õppeasutusele ja kasutajal on privileeg VIEW_CERTIFICATES, kasutaja rolliks väljundisse määratakse role.base = "ISSUER".
b.	**Teenus tuvastab kasutajal õiguse lõpudokumendi andmeid pärida isikukoodile antud kehtiva ligipääsu alusel.** Lõpudokumendi ligipääsude seas leidub kehtivaid ligipääse (Access.status = "VALID", Access.endDate >= SYSDATE) tüübiga "ID_CODE", mille ligipääsukood (Access.accessorCode) vastab sisse loginud kasutaja isikukoodile, kasutaja rolliks väljundisse määratakse role.base = "USER".

c.	**Teenus ei tuvasta kasutajal õigust andmeid pärida ning tagastab vea.** Vea kood 001, tunnistuse andmeid ei tagastata.
3.	**Teenus tagastab lõpudokumendi üldandmed, sellega seotud dokumentide metaandmed ning kasutaja rolli andmete pärimisel.** Üldandmed andmeobjektist ET Index, tunnistusega seotud dokumentide andmed andmeobjektist ET Document, kasutaja rolli andmed eelmises sammus tuvastatud õiguse järgi.
a.	**Kui andmeid päritakse ligipääsu põhjal, mille ulatuseks määratud ainult põhidokument, siis tagastatakse ainult lõpudokumendiga seotud põhidokumentide andmed.** Ligipääsu ulatuse Access.scope = "MAIN_DOCUMENT" puhul tagastatakse ainult need dokumendid, mille Document.type = Index.type.

3.4 Järeldused ja tulevik

Üldhariduse lõpudokumentide teenuse disaini tulemusel võib järeldada, et antud töö käigus defineeritud mikroteenuste disaini ja dokumenteerimise põhimõtteid saab kasutada mikroteenuste arhitektuuril põhineva süsteemi kavandamiseks ja kirjeldamiseks riigisektoris. Üldhariduse lõpudokumentide analüüsis kirjeldatud lahenduse abil on tänaseks teostatud projekti skoobis olev tagasüsteemi funktsionaalsus - see tähendab EHISe lõpudokumentide *API* esmane versioon.

Mikroteenuste disaini põhimõtete defineerimisel ja nende rakendamisel riigisektoris ei tulnud käesoleva töö käigus välja sellele sektorile ainuomaseid erisusi.

Diplomitöös kirjeldatud disaini põhimõtteid saab seniste tulemuste põhjal kasutada ka tulevaste EHIS2 funktsionaalsuste välja töötamisel. Kuna EHIS2 projekti raames on plaanis hakata järk-järgult funktsionaalsust EHISe vanast osast uude üle viima, siis tuleks tulevaste tööde puhul täiendavalt uurida erinevaid strateegiaid monoliitsest süsteemist funktsionaalsuse mikroteenustele üle viimiseks.

Mikroteenuste dokumentatsiooni olulisim osa kasutajate jaoks on teenuste liideste dokumentatsioon. Üldhariduse lõpudokumentide teenuse disainil joonistus välja, et see dokumentatsioon ei vasta alati teenuse kasutaja vajadustele. Seetõttu võeti paralleelselt kasutusele automaatne teenuse liideste dokumenteerimine. Tulevikus tuleks põhjalikumalt uurida liideste dokumentatsiooni kvaliteedi parandamise võimalusi. Selleks võiks uurida lähemalt erinevaid avatud standardeid ning automaatlahendusi liideste dokumenteerimiseks ning kaaluda erinevate lahenduste eeliseid ning puuduseid. Automaatse dokumenteerimise tööriistadest mõned pakuvad ka lahendusi, kus standardselt modelleeritud liidese kirjelduse põhjal saab automaatselt genereerida koodi. Sellise strateegia kasutamise efektiivsust, kus teenuse liidesed kirjeldatakse algusest peale standardset notatsiooni kasutades, tuleks samuti uurida.

Praeguses faasis on EHIS2 süsteemis asuvate teenuste hulk veel väike, kuid aja jooksul see järjest kasvab. Tulevikus on vaja süsteemist tekitada ka komponentide ülene vaade – nö suur pilt, et mitte kaotada ülevaadet süsteemist tervikuna.

4 Kokkuvõte

EHISe üldhariduse lõpudokumentide registri lahenduse täiendamise projekt otsustati teostada EHIS2 süsteemis, kasutades mikroteenuste arhitektuuri. See oli üks esimesi komponente, mis EHISe süsteemis sellise arhitektuurilise lähenemise abil otsustati teostada ning seoses sellega tekkis vajadus tehnilise spetsifikatsiooni loomise vaatepunktist selgitada välja mikroteenuste kavandamisega seotud põhimõtted ja nõuded nende dokumenteerimisele.

Käesoleva diplomitöö põhieesmärgiks oli luua näidis mikroteenuste kavandamisest ja kirjeldamisest riigisektoris EHISe üldhariduse lõpudokumentide registri uue lahenduse näitel. Põhieesmärki toetavaks eesmärgiks oli sõnastada mikroteenuste disaini ja dokumenteerimise põhimõtted selliselt, et neid oleks võimalik võtta aluseks tulevastes projektides disaini otsuseid tehes.

Mikroteenuste disaini ja dokumenteerimise põhimõtete kirjeldamiseks uuris autor mikroteenuste arhitektuuri omadusi võrreldes monoliitse arhitektuuri ning teenusepõhise arhitektuuriga ning tõi selle võrdluse tulemusena välja mikroteenuste eelised ja puudused. Mikroteenuste eelistest ja puudustest tuletati mikroteenuste disainil olulised põhimõtted. Nendest põhimõtetest olulisemana võib välja tuua komponentide ehk teenuste piiritlemise ärilise valdkonna järgi. Lisaks on oluline põhimõte, et komponentide avalik info ehk liidesed tuleks disainida lähtuvalt kasutajate vajadustest. Tähelepanu tuleks pöörata ka sellele, et teenuse sisemine osa – andmete säilitamise ja töötlemise reeglid – jääksid kasutajatele peidetuks.

Dokumenteerimisel toodi olulise põhimõttena välja, et dokumentatsioon peaks peegeldama teostatavat lahendust nii sõnastuse kui ka struktuuri poolest ning selleks tuleb dokumentatsiooni jooksvalt värskendada. Liideste dokumenteerimisel tuleb silmas pidada, et selle dokumentatsiooni sihtrühmaks on ka liidese kasutajad. Teenuse sisemise osa dokumentatsiooni loomisel tuleks lähtuda ärivaldkonna spetsiifikast.

Lõpuks uuriti disaini ja dokumenteerimise erisusi riigisektoris, kus ei joonistunud välja suuri erisusi võrreldes üldiste põhimõtetega.

Kolmandas peatükis tehti ülevaade disaini ja dokumenteerimise põhimõtete rakendamisest üldhariduse lõpudokumentide teenuse kavandamisel EHISes ning toodi näiteid tekkinud raskustest ning probleemides. Detailsem näide toodi lõpudokumendi päringu *GET /certificate/{indexId}* liidese ja andmete töötlemise dokumentatsioonist.

Järeldustena tehti kokkuvõte üldhariduse lõpudokumentide disainist ja varasemalt sõnastatud põhimõtete rakendamise tulemustest. Tulevaste uurimisteemadena soovitati uurida monoliitse süsteemi mikroteenustele üle viimise strateegiaid, liideste dokumentatsiooni kvaliteedi tõstmise lahendusi ning mikroteenuste arhitektuuri puhul süsteemist ülevaate säilitamise meetodeid.

Kasutatud kirjandus

Evans, Eric. *Domain-driven design.* 2003.

Fowler, Martin. *Microservice Trade-Offs.* 1. juuli 2015. a. https://martinfowler.com/articles/microservice-trade-offs.html.

—. *MonolithFirst.* 3. juuni 2015. a. https://martinfowler.com/bliki/MonolithFirst.html (kasutatud 25. aprill 2020. a.).

Fulton, Scott. *The New Stack.* 8. oktoober 2015. a. https://thenewstack.io/led-amazon-microservices-architecture/.

Haridus- ja Teadusministeerium. „EHIS2 raamhange.“ 27. 09 2018. a. https://riigihanked.riik.ee/rhr-web/#/procurement/727579/documents/source-document?group=B&documentOldId=11994783 (kasutatud 7. aprill 2020. a.).

Iaboni, Dan. *SCENARIOS, USER STORIES AND USE CASES…OH MY!* 21. november 2013. a. https://www.akendi.com/blog/scenarios-user-stories-and-use-casesoh-my/ (kasutatud mai 2020. a.).

Ismail, Kaya. *CMSWire.* 23. august 2018. a. https://www.cmswire.com/information-management/7-tech-giants-embracing-microservices/.

Johnson, Tom. *Documenting API endpoints.* 2019. https://idratherbewriting.com/learnapidoc/docendpoints.html (kasutatud mai 2020. a.).

Kononow, Piotr. *3 Basic Data Modeling Techniques - ERD, UML and Data Dictionary.* 13. november 2019. a. https://dataedo.com/blog/basic-data-modeling-techniques (kasutatud mai 2020. a.).

Kütt, Andres. „API First Government.“ 2016. https://www.slideshare.net/AndresKtt/api-first-government (kasutatud 19. aprill 2020. a.).

Lewis, James, ja Martin Fowler. *Microservices.* 25. märts 2014. a. https://martinfowler.com/articles/microservices.html (kasutatud 18. aprill 2020. a.).

Newman, Sam. *Building Microservices.* O'Reilly Media, Inc., 2015.

Riigi Teataja. „Põhikooli ja gümnaasiumi lõputunnistuse ning riigieksamitunnistuse statuut ja vormid.“ 17. veebruar 2018. a. https://www.riigiteataja.ee/akt/106052017003?leiaKehtiv (kasutatud 13. aprill 2020. a.).

Trinidad Wiseman OÜ. „Elektroonsete kutse- ja lõputunnistuste ärianalüüs.“ 2018. https://pilv.hm.ee/index.php/s/oi8h3EMEf091PLk?_ga=2.261434189.1900976663.1580932420-972481338.1534155813 (kasutatud 10. 04 2020. a.).

Vabariigi Valitsus. „Eesti infoühiskonna arengukava 2020.“ 2018. https://www.mkm.ee/sites/default/files/eesti_infouhiskonna_arengukava.pdf (kasutatud 22. aprill 2020. a.).

Vaher, Kristo. „Next Generation Digital Government Architecture.“ 2020. https://46lsnmttuzs3omis7vq45py1-wpengine.netdna-ssl.com/wp-content/uploads/2020/03/Next-Generation-Digital-Government-Architecture-1.0.pdf (kasutatud 17. aprill 2020. a.).

Wiegers, Karl E, ja Joy Beatty. *Software Requirements.* Microsoft Press, 2013.

Lisa 1 – *Document* andmeolemi kirjelduse näide

Selgitus:	Lõpudokumendi sisudokument. Ühe lõpudokumendiga võib olla seotud mitu sisudokumenti: nt lõputunnistus ja hinneteleht. Lõpudokumendi sisudokumenti muudatusi tehes salvestatakse alati dokumendi sisust uus redaktsioon. Dokumendi sisu eelmised redaktsioonid säilivad.	
Atribuudi nimetus	**Tüüp, kohustuslikkus, piirangud**	**Sisuline selgitus**
type	Võimalikud väärtused: • BASIC_EDUCATION_CERTIFICATE • BASIC_EDUCATION_TRANSCRIPT_OF_GRADES • GENERAL_EDUCATION_CERTIFICATE • GENERAL_EDUCATION_TRANSCRIPT_OF_GRADES	Näidatakse dokumendi sisu liigi klassifikaatori koodi.
typeName	Võimalikud väärtused: • "Põhikooli lõputunnistus" • "Põhikooli hinneteleht" • "Gümnaasiumi lõputunnistus" • "Gümnaasiumi hinneteleht"	Näidatakse dokumendi sisu liigi klassifikaatori nimetust.
number		Redaktsioonil kuvatav dokumendi number.

status	Võimalikud väärtused: • VALID • INVALID	Näidatakse kas dokumendi sisu on kehtiv ("VALID") või kehtetu ("INVALID").
revision	number	Redaktsiooni number
content		Sisu salvestatakse JSON formaadis.
language		Näidatakse dokumendi sisu keelt. EHIS2 keelte klassifikaator.
signatureUID		Viide räsimisel saadud signatuurile.