TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Robert Samuel Roos 193514IAIB

# TEENUSE KIHI REALISEERIMINE UUDE DEKANAADI TÖÖTAJATE ÕPPEINFOSÜSTEEMI EHISE DIGITAALSETE LÕPUDOKUMENTIDE SAATMISE NÄOL

Bakalaureusetöö

Juhendaja: Evelin Halling, PhD
Kaasjuhendaja: Irina Kelder

Tallinn 2024

# 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: Robert Samuel Roos

09.05.2024

# Annotatsioon

Uus Tallinna Tehnikaülikooli dekanaadi töötajatele mõeldud õppeinfosüsteem on projekt, mida arendab firma Fujitsu AS Estonia.

Antud bakalaureuse töö eesmärk on leida lahendus andmevahetuse teenuse kihi realiseerimiseks uude õppeinfosüsteemi lõpudokumentide Eesti Haridus Infosüsteemi (EHIS) edastastamise näitel.

Uuel õppeinfosüsteemil on andmevahetuse vajadus kolmandate süsteemide ja rakendustega, milleks antud bakalaureusetöös on Eesti Hariduse Infosüsteem.

Bakalaureusetöö tegemise käigus analüüsiti erinevaid teenuse kihi arhitektuure ja stuktuure. Lisaks analüüsiti ka lõpudokumentide Eesti Haridus Infosüsteemi edastamise funktsionaalsust ja tingimusi.

Koostatud analüüse kasutades realiseeriti parim leitud teenuse kihi struktuur arendades lõpudokumentide edastamise funktsionaalsust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 6 peatükki, 19 joonist, 0 tabelit.

# Abstract

## Realization of the Service Layer in the Form of Sending EHIS Graduation Certificates for the New Study Information System Intended for the Employees of the Dean's Office

The new study information system intended for the employees of Tallinn's University of Technology dean's office is a projekt, which is developed by the company Fujitsu AS Estonia.

The purpose of this bachelor's thesis is to find a solution for the realization of the data exchange service layer in the form of sending graduation certificates to the Estonian Education Information System for the new infomation system (EHIS).

The new educational information system for the employees of the dean's office has the need to exchange data with thirdparty systems and application, one of which in this bachelor's thesis is the Estonian Education Information System.

Different structures and architectures of the service layer were analyzed during this bachelor's thesis. Futhermore, the functionality and conditions for transferring graduation documents to the Estonian Education Information System were also analyzed.

Using the compiled analyses, the best found service layer structure was realized by developing the funtionality of transferring graduation documents.

The thesis is in Estonian and contains 29 pages of text, 6 chapters, 19 figures, 0 tables.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| EHIS | Eesti Hariduse Infosüsteem |
| ÕIS | Õppeinfosüsteem |
| WSDL | Web Service Description Language |
| SPA | Single Page Application |
| CRUD | Create, Read, Update, Delete |
| Uus ÕIS | Uus dekanaadi töötajatele mõeldud õppeinfosüsteem |
| ois-ws | Uus andmevahetuse teenuse kiht, mis hakkab kolmandate rakendustega suhtlema |
| PDF | Portable Document Format |
| Akad. Õiend e.k. | Eesti keelne akadeemiline õiend |
| Akad. Õiend i.k. | Inglise keelne akadeemiline õiend |
| REST | Representational state transfer |
| SOAP | Simple Object Access Protocol |
| API | Application Programming Interface |
| TalTech | Tallinna Tehnikaülikool |
| DAO | Data Access Object |
| DTO | Data Transfer Object |
| SQL | Structured Query Language |

# Sisukord

# Jooniste loetelu

# 1 Sissejuhatus

Uus TalTech-i dekanaadi töötajatele mõeldud õppeinfosüsteem on projekt, mida arendab firma Fujitsu Estonia AS. Antud uus õppeinfosüsteem on kaasaegne TalTech-i ametliku visuaalse stiiliga rakendus, mis on arendatud kaasaegsetel tehnoloogiatel.

Uus õppeinfosüsteem on arendatud, et tagada dekanaadi töötajatele lihtne, mugav ja efektiivne keskkond enda tööülesannete täitmiseks.

## 1.1 Taust ja probleem

Uuel TalTech-i dekanaadi töötajatele mõeldud õppeinfosüsteemil tuleb hakata andmevahetust teostama kolmandate süsteemide ja rakendustega. Üheks selliseks kolmandaks süsteemiks on Eesti Haridus Infosüsteem (EHIS), kuhu on tarvis edastada digitaalsed lõpudokumendid kasutades *X-teed* [1] ja *SOAP* protokolli. Lisaks lõpudokumentide edastamisele kasutades *SOAP* protokolli peab olema võimalik andmevahetus teiste kolmandate süsteemidega kasutades *REST API* arhitektuuri.

Hetkel puudub uues dekanaadi õppeinfosüsteemis vastav andmevahetuse teenuse kiht.

## 1.2 Lõputöö eesmärk

Käesoleva bakalaureusetöö ülesanne on analüüsida erinevaid teenuse kihtide struktuure ning arhitektuure ja realiseerida valitud parima struktuuriga andmevahetuse teenuse kiht.

Antud kiht realiseeritakse uude dekanaadi töötajatele mõeldud õppeinfosüsteemi digitaalsete lõpudokumentide EHIS-esse saatmise näitel.

Realiseeritud teenuse kiht peab olema lihtsasti ja mugavalt hallatav arenduste ja paranduste teostamiseks. Andmevahetuse teenuse kiht peab olema võimeline kasutama *SOAP* protokolli, et oleks võimalik edastada lõpudokumente EHISesse, ja *REST API* arhitektuuri, et oleks võimalik andmevahetus süsteemide ja rakendustega, mis kasutavad *REST API* arhitektuuri.

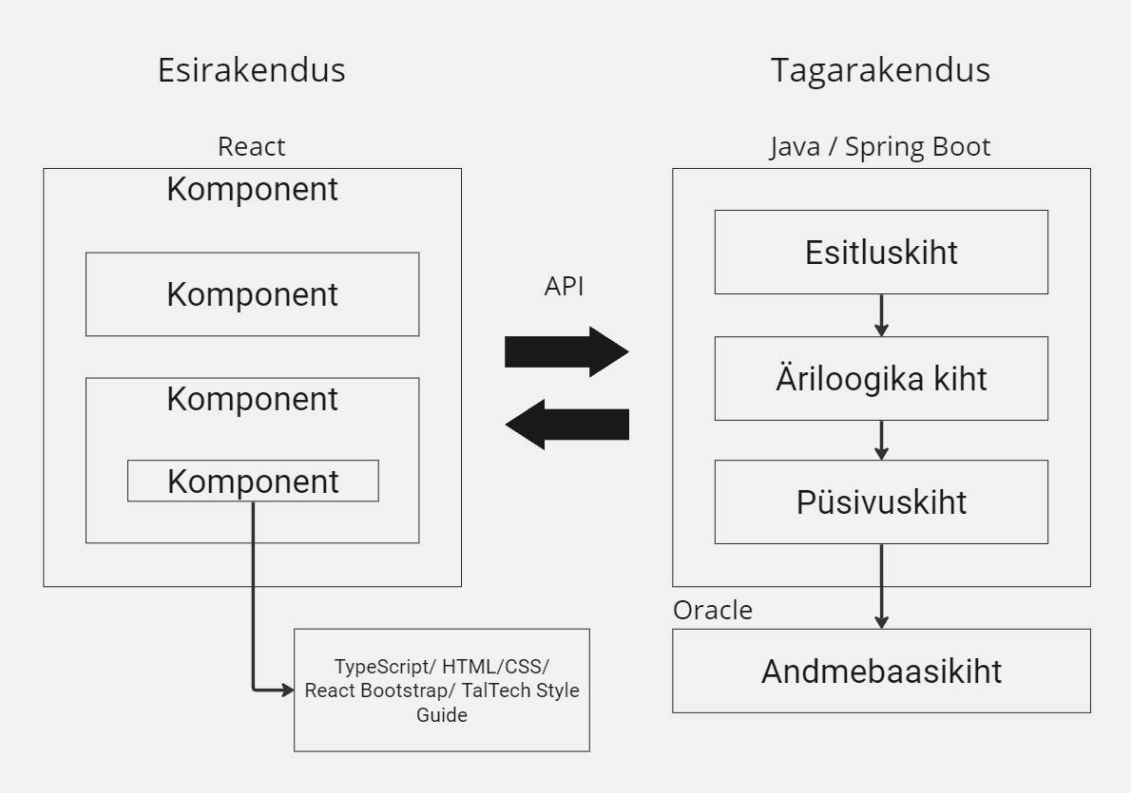
### 1.3 Lõputöö metoodika

Antud teenuse kihi realiseerimiseks teostatakse analüüs erinevate teenuse kihi struktuuridest küsitledes autori tiimis olevaid kaasarendajaid nende kogemuste ja arvamuste jaoks ning uurides internetist erinevaid it projektide arhitektuurilisi lahendusi. Teenuse kihi analüüsist valitakse parim lahendus antud tingimustest lähtudes, mida hakatakse realiseerima lõpudokumentide EHISesse saatmise näol.

Lõpudokumentide EHISesse saatmise funktsionaalsuseks teostab autor koos tiimi analüütikuga analüüsi. Analüüsitakse EHISe *X-tee* [1] teenusid, mida kasutades hakatakse realiseerima lõpudokumentide saatmise funktsionaalsust uude ÕIS-i projekti.

# 2 Uue ÕIS-i projekti tehnoloogiad

Esirakenduse ja tagarakenduse vaheline suhtlus käib läbi *REST API*. (Joonis 1).




Joonis 1. Uue ÕIS-i lihtsustatud arhitektuuri joonis

Alampunktides on kirjeldatud lühidalt uue ÕIS-i projekti esirakenduse ja tagarakenduse tähtsamaid tehnoloogiaid.

## 2.1 Esirakendus (front-end)

Dekanaadi töötajatele mõeldud õppeinfosüsteemi esirakendus on arendatud TypeScript programmeerimis keeles kasutades React raamistikku. Tähtsamad tehnoloogiad on välja toodud alampeatükkidena.

### 2.1.1 React

React on JavaScripti raamistik, mis lihtsustab luua interaktiivseid kasutajaliideseid, tagades kiiruse, lihtsuse ja skaleeritavuse. React on kasulik raamistik üheleherakenduste (SPA) ehitamiseks. [2]

### 2.1.2 TypeScript

TypeScript on programmeerimiskeel, mis põhineb JavaScriptil täiendades seda. TypeScripti peamine eelis on koodis ootamatu käitumise esile toomine, mis vähendab vigade tõenäosust. [3]

### 2.1.3 React Bootstrap

React Bootstrap on stiilide teek, mis asendab JavaScripti Bootstrapi. Iga komponent on nullist üles ehitatud Reacti komponendina ilma ebavajalike sõltuvusteta. React Bootsrap on üks vanimatest Reacti teekidest, mis on Reactiga koos arenenud, mistõttu on ta suurepärane valik Reacti projketi kasutajaliidese alusena. [4]

### 2.1.4 TalTech Style Guide

TalTech Style Guide on Tallinna Tehnikaülikooli poolt kokku pandud stiiliraamat ja React komponentide teek. TalTech Style guide-i komponendid on ehitatud React Bootstrapi komponentide pealt ning teevad arendaja tööd oluliselt mugavamaks.

### 2.1.5 HTML

HTML (HyperText Markup Language) on kood, millega ehitatakse ja struktureeritakse veebileht ja selle sisu. [15]

### 2.1.6 CSS

CSS (Cascading Style Sheets) on laaditabeli keel, millega vormistatakse veebileht. CSS kirjeldab, kuidas elemente tuleks renderdada ekraanil. [16]

## 2.2 Tagarakendus (back-end)

Dekanaadi töötajatele mõeldud õppeinfosüsteemi tagarakendus on kirjutatud Java 17 programmeerimis keeles ja on üles ehitatud Spring Boot raamistikule. Tähtsamad tehnoloogiad on välja toodud alampeatükkidena.

### 2.2.1 Java

Java on kõrgetasemeline, klassipõhine, objektorienteeritud programmeerimiskeel. [5]

### 2.2.2 Spring Boot

Spring Boot on osa Spring Java raamistikust. Spring Boot lihtsustab Java põhise rakendusega töötamist. [6]

### 2.2.3 Gradle

Gradle on rakenduse ehitamise automatiseerimise tööriist, millega on võimalik arendaja mitmetes erinevates keeltes tarkvara. Gradle on väga kasulik tööriist, mis toetab tarkvara koostamise protsessi kõikides etappides, sealhulgas kompileerimine, kontrollimine, sõltuvuse lahendamine, testi täitmine, lähtekoodi genereerimine, pakkimine ja avaldamine. [7]

### 2.2.4 Andmebaas

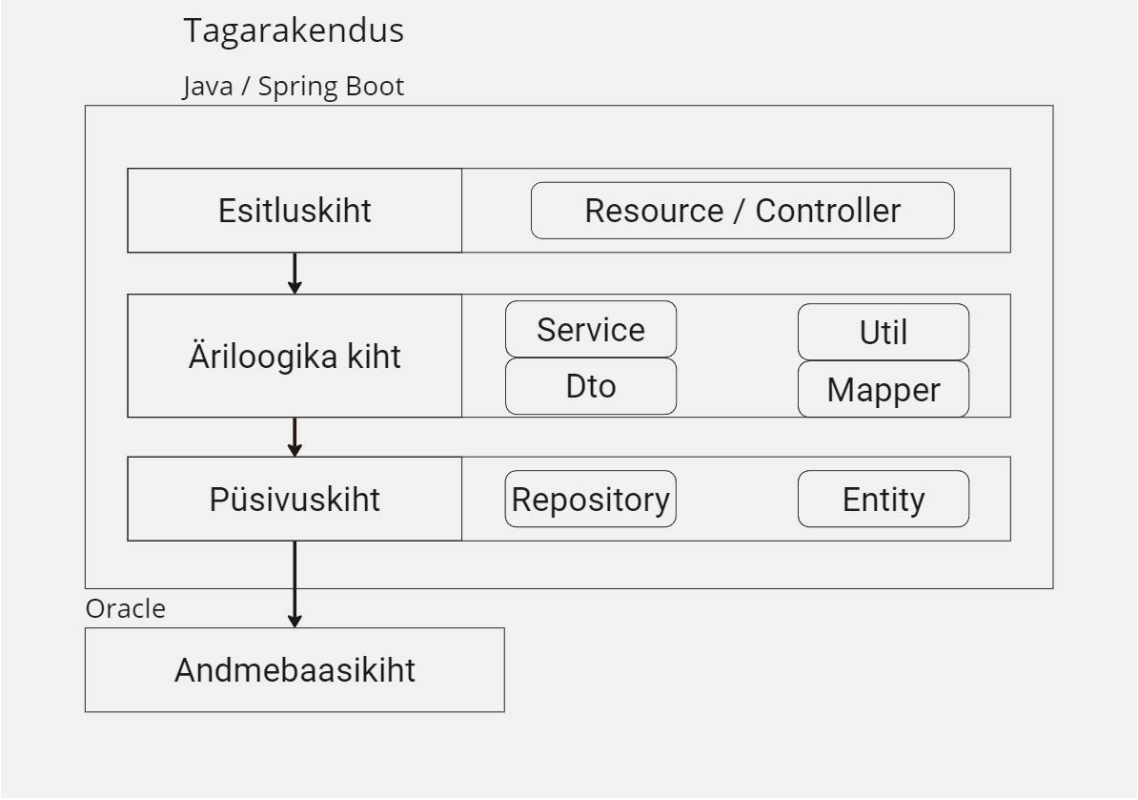
Dekanaadi ÕIS kasutab vana õisiga ühist andmebaasi. Antud andmebaas on Oracle andmebaaasisüsteemil.

# 3 Teenuse kiht

Kõige tavalisem ahritektuurimuster, mis on enamiku Java rakenduste standard mustreid, on kihiline arhitektuurimuster. Seda tuntakse ka n-tasandi arhitektuurimustrina.[8] [9]

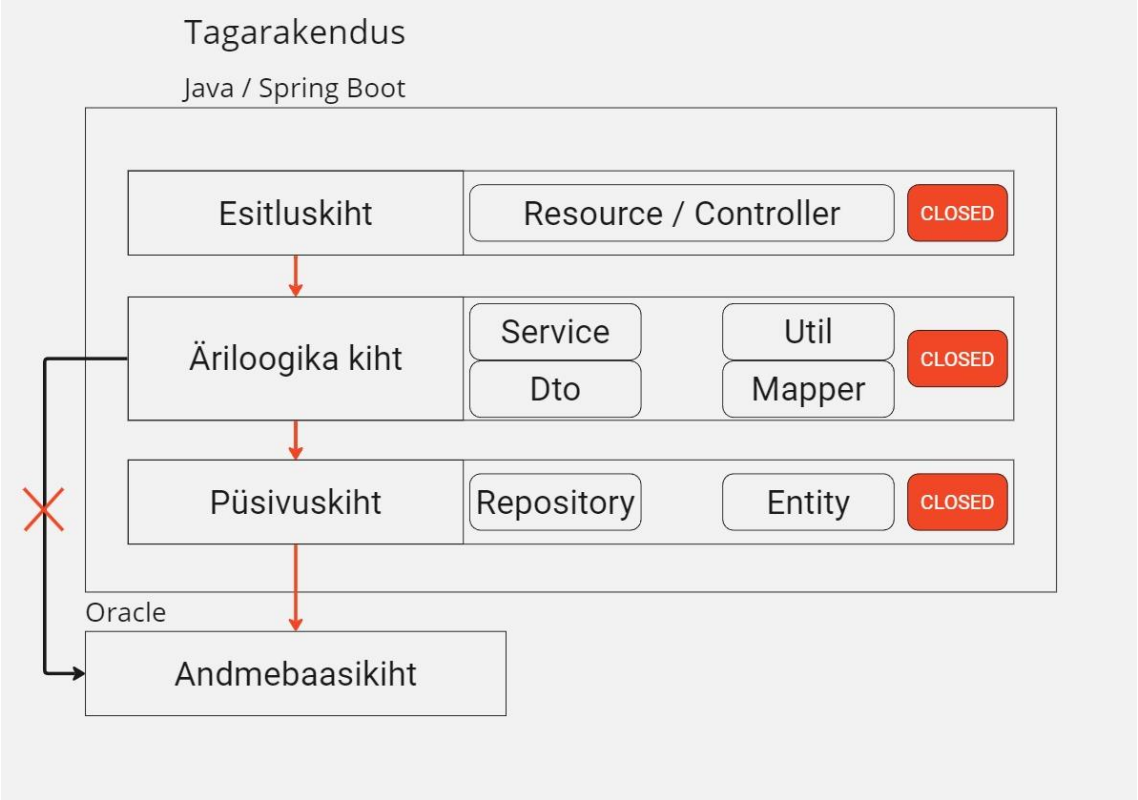
Kihilise mustriga rakendused on korraldatud horisontaalseteks kihtideks, mis esindavad tarvara erinevaid tasemeid ja tüüpe. Antud kihid aitavad rakenduse jagada paremini hallatavateks üksusteks. Jagatud kihid suhtlevad endaga samal tasemel või selle all oleva taseme kihtidega ja igal kihil on kindel ülesanne, vastutus ja roll, mida antud kiht rakenduses täidab ning millest ta väljapoole ei liigu. [8] [9]

Kuigi kihilise arhitektuuri muster ei määra mustris kindlat kihtide arvu ja tüüpe, koosnevad enamik kihilistest arhitektuuridest neljast sandardkihist: Esitluskiht, äriloogika kiht, püsivuskiht ja andmetekiht. [9] (Joonis 2)




Joonis 2. Kihiline arhitektuur

Suletud ehk kinniste kihtidega kihiline arhitektuur tähendab seda, et ühelt kihilt teisele liikudes ei ole päringul võimalik hüpata üle antud kihi all olevast kinnisest kihist ning peab selle läbima, et liikuda edasi järgmiste kihtide juurde. [8] [9] (Joonis 3)




Joonis 3. Kinniste kihtidega kihiline arhitektuur

Alampeatükkides on igat kihti lühidalt kirjeldatud.

### 3.1.1 Esitluskiht

Esitluskiht on kõige kõrgem kiht ja selle kihi vastutus on suhtlus kasutajaliidesega ja sisaldab ainult andmete esitamise ja CRUD (create, read, update, delete) operatsioonide alguse loogikat. Esitluskihti paigutatakse projekti *Controller* või *Resource* klassid, mis on seotud projekti *API* päringute lõpp-punktidega, kuhu esirakendus päringuid saadab. Selle kihi loogika peaks olema kõige lihtsam. [8] [17]

### 3.1.2 Äriloogika kiht

Äriloogika kiht sisaldab suuremat osa rakenduse loogikast. Antud kiht suhtleb nii esitluskihi kui ka püsivuskihiga. Äriloogika kiht sisaldab projekti *Service*, *Mapper*, *Util* ja vajadusel muid klasse, mis tegelevad projekti andmete töölemisega. [8]

*Service* klassid realiseerivad projekti andmete töötlemise loogika. Seda loogikat soodustavad *Util* klassid, mis koosnevad enamasti lühikestest funktsioonidest. Antud funktsioone on võimalik korduvalt kasutada andmete töötlemise kergendamiseks. [8]

Lisaks sisaldab antud kiht ka *DTO* (Data Transfer Object) klassi, mida kasutades edastatakse esirakendusele andmeid. *DTO* klasse on võimalik koostada *Entity* klassist (ja vastupidi) kasutades *Mapper* klassi. [8]

Äriloogika kihti on võimalik jagada veel väiksemateks loogika kihtideks vastavalt vajadusele. [8]

### 3.1.3 Püsivuskiht.

Püsivus- või andmejuurdepääsukiht on kiht, mille ülesanne on luua side andmeallikaga, milleks tavaliselt on andmebaas. Siin kihis koostatakse päringud andmete töötlemiseks. Siia kihti paigutatakse *Repository* ja *Entity* klassid. [8] [9]

*Repository* klass koosneb päringutest, mis kasutavad SQL (Stuctured Query Language) programmeerimis keelt andmebaasiga suhtlemiseks ja mis tagastavad *Entity* klasse.

*Entity* klassi objekt on andmebaasi tabeli kirje objekti näol.

### 3.1.4 Andmebaasikiht

Andmebaasikiht on viimane kiht, mis koosneb ühest või mitmest andmebaasist. Siin kihis käivitatakse ka CRUD operatsioone ja püsivuskihis koostatud päringuid. [9]

## 3.2 Teenuse kihi analüüs

Uue ÕIS-i arhitektuur kasutab ka kihilist arhitektuurimustrit, milles on mitmeid kihte, kuid antud bakalaureusetöö jaoks kasutatakse lihtsustatud uue ÕIS-i tagarakenduse arhitektuuri pilti. (Joonis 4)


Joonis 4. Lihtsustatud uue ÕIS-i andmevahetuse voo näide

Selleks, et antud teenuse kiht realiseerida, on tarvis analüüsida ning otsustada kuhu ja kuidas paigutada vajalik andmevahetuse teenuse kiht. Lisaks on tarvis sinna struktureerida vajalikud failid, klassid ja loogika nii, et teenuse kiht oleks võimalikult arusaadav ning vajadusel lihtsasti hallatav.

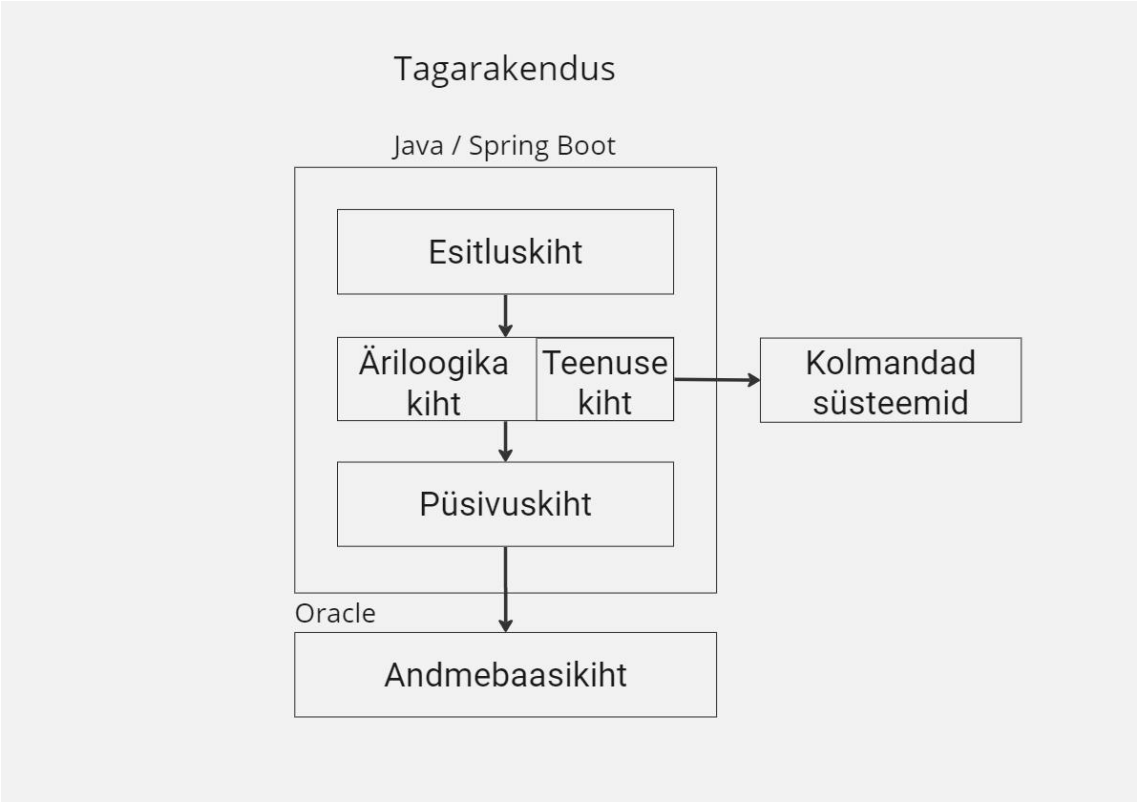
Uue ÕIS-i äriloogika kiht hakkab kasutama antud teenuse kihti kolmandate osapooltega (antud juhul EHIS) suhtlemiseks. Töö eesmärgiks on välja mõelda, kuidas ehitada *ois-ws*-i struktuur nii, et ta oleks võimalikult efektiivne andmevahetuses, kasutades nii *SOAP* protokolli kui ka *REST* arhitektuuri, ja lihtsasti arusaadava struktuuriga, et teda oleks vajadusel võimalikult lihtne ja mugav parendada, edasi arendada ja hallata.

Kuna autoril puudub varasem kogemus teenuse kihtidega, alustati teenuse kihi struktuuri ja arhitektuuri analüüsiks kaastöötajate ja teiste arendajatega küsitlemisega nende kogemuste ja arvamuste jaoks, kust leiti välja neli võimalikku teenuse kihi struktuuri: Uue ÕIS-i äriloogika kihti paigutamine, uue ÕIS-i projektist eraldi projektina arendamine, Uue ÕIS-i projektis lahtise kihina arendamine ja uue ÕIS-i projektis eraldi kihina arendamine.

Alampeatükkides on kirjeldatud rohkem leitud teenuse kihi paigutamise lahendustest ning nende analüüsimisest

### 3.2.1 Uue ÕIS-i äriloogika kihti paigutamine

Esimesene lahendus tekkis küsimusest: „Miks mitte lihtsalt andmevahetuse kihi loogika lisada meie projekti äriloogika kihti?“ ehk jagada vajalik loogika ja vastavad klassid ja failid uue ÕIS-i äriloogika kihti. (Joonis 5)




Joonis 5. Lihtsustatud uue ÕIS-i andmevahetuse voo näide kui teenuse kiht paigutatakse äriloogika kihti

Algul tundub antud lahendus lihtne ja loogiline, et paigutada uus vajalik loogika ja antud loogikas kasutatavad klassid põhirakenduse äriloogika kihti sama struktuuri järgi nagu seal hetkel loogika, klassid ja failid on jaotatud.

Kui uue ÕIS-i projekti ja selle struktuuri ning ois-ws teenuse kihi tulevast ülesannet (Vaata punkt 4), siis on näha antud lahenduse miinuseid.

Uus ÕIS-i projekt on suur ja keeruline süsteem, mis koosneb paljudest *API* päringute lõpp-punktidest, keerukast ja suurest loogikast ning väga paljudest andmetest. Hetkel

kogu loogika uue ÕIS-i äriloogika kihis on arendatud ja töötab rakenduse enda andmete töötlemiseks esirakendusega suheldes.

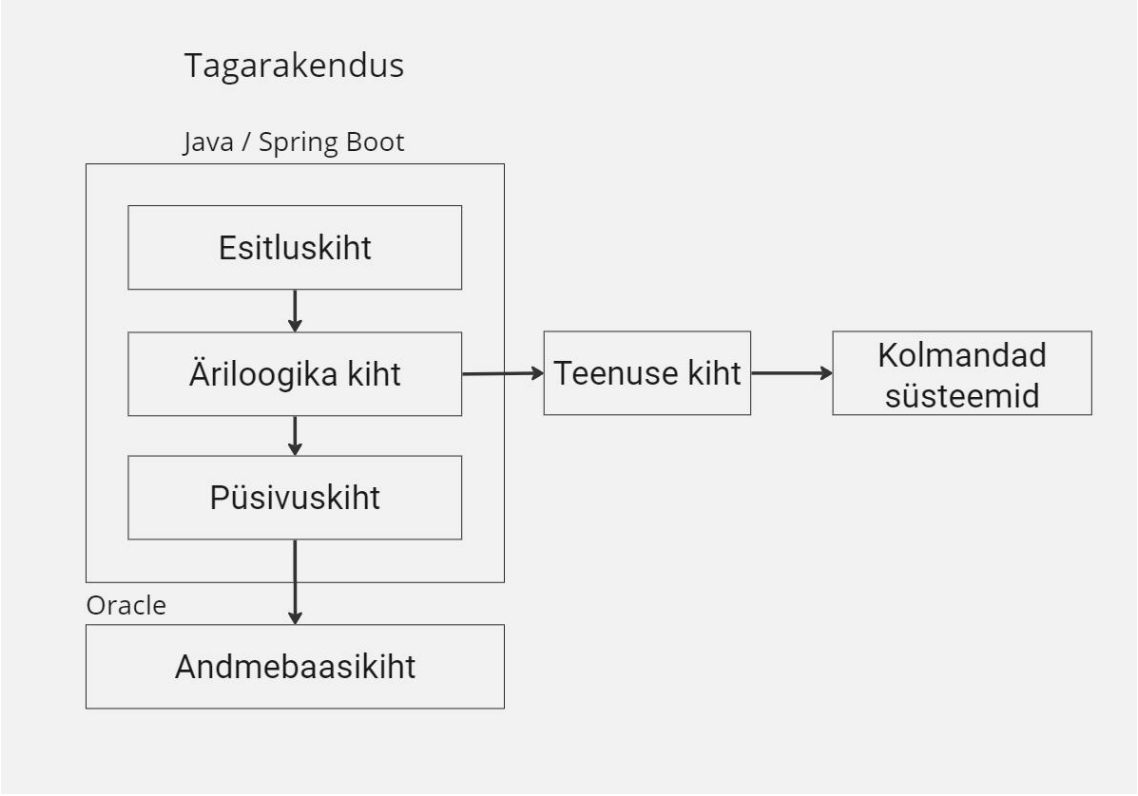
Kui uue ÕIS-i äriloogika kihti lisada veel *ois-ws* andmevahetuse teenuse kihi vajalikud klassid ja loogika, mis ei ole otseselt seotud uue ÕIS-i kasutajaliidesega vaid võimaldab suhtlust kolmandate rakenduste ja süsteemidega, siis muutub antud äriloogika kiht ebavajalikult keeruliseks ja segaseks.

Isikutel ja arendajatel, kes ei ole ise tegelenud *ois-ws* andmevahetuse loogikaga, on keeluline teenuse kihti hallata. Neil on rakse leida ja eristada faile ja loogikat, mis on osa *ois-ws* teenuse kihist, paljude uue ÕIS-i failide ja kataloogide seast.

Autor ei välista, et rohkema analüüsiga võib olla võimalik leida antud lahendusest struktuuriline versioon, mis töötab ja sobiks, kuid hetkel on leitud paremaid lahendusi

### 3.2.2 Uue ÕIS-i projektist eraldi projekti/rakenduse arendamine

Teiseks leitud lahenduseks on *ois-ws* andmevahetuse teenuse kihi arendus eraldi projektina. (Joonis 6)




Joonis 6. Lihtsustatud uue ÕIS-i andmevahetuse voo näide kui teenuse kiht arendatakse eraldi projektina

Teenuse kihi arendamine uuest ÕIS-i projektist eraldi projektina on lahendus, mille pakkusid välja kaasarendajad, sest sellise struktuuriga on neil kogemust.

Siin on mõeldud, et andmevahetuse klassid ja loogika arendatakse eraldi projkektina, mis siis kasutamiseks kompileeritakse jar failiks ning antud jar fail lisatakse uude ÕIS-i projekti, kus äriloogika kiht hakkab siis *ois-ws* teenuse kihi meetodeid ja funktsionaalsust kasutama.

Antud lahenduse eelised on haldamine. Kuna teenuse kiht on täiesti eraldi projektina arendatud uue ÕIS-i projektist, kaob segadus uue ÕIS-i ja *ois-ws* teenuse kihi failide ja loogika eristamisel. Kui *ois-ws* teenuse kihti on tarvis parandada, parendada või uuendada, siis on lihtne ette võtta antud *ois-ws* teenuse kihi eraldi arendatud projekt, seal vajalikud muudatused teostada, uuesti kompileerida ja kasutavasse projekti uuesti lisada.

Antud lahenduse miinuste hulka kuulub ka haldamine. Kui *ois-ws* teenuse kihs on tarvis palju muudatusi ja arendusi teostada ning kui antud teenuse kihi kallal töötavad mitmed arendajad korraga, siis eraldi projektina tuleb teenuse kihi haldamisega kaasa ebamugavad lisa sammud. Seda esineb palju just arenduse käigus, näieks pärast iga arendaja muudatust, parandust või arendust tuleb uuesti kompileerida ning kompileeritud jar fail projekti lisada nii, et kõik arendajad saaksid ka viimase versiooni teenuse kihist ning et see nendel midagi katki ei teeks.

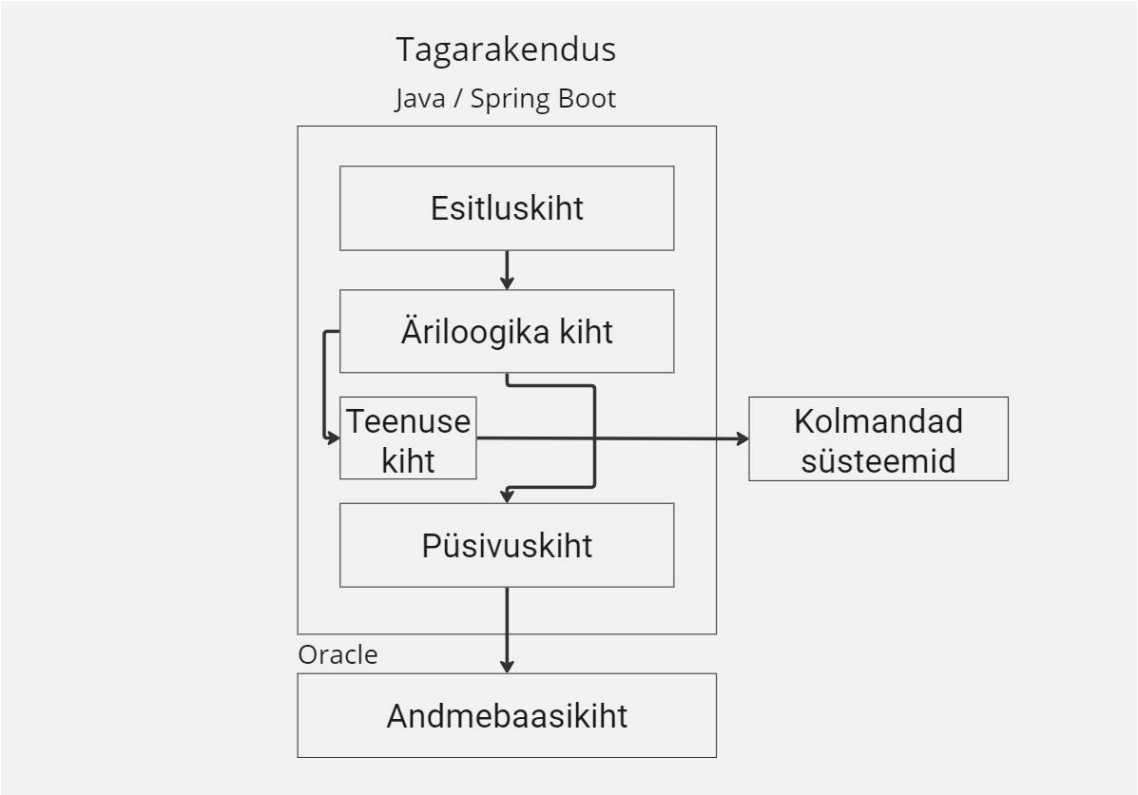
Lisaks kui teenuse kiht ei ole liiga suur ja/või seda ei kasuta rohkem kui üks rakendus, siis kas antud ebamugavad lisasammud on seda väärt ning kas on vajadus arendada antud teenus täiesti eraldi projektina.

Kuna hetke seisuga ei muutu *ois-ws* teenuse kiht üleliia suureks ning seda ei hakka hetke seisuga kasutama ükski teine projekt peale uue ÕIS-i projekti, otsustati mitte kasutada antud lahendust, vähemalt praeguse analüüsi ja arendamise käigus.

### 3.2.3 Uue ÕIS-i projektis lahtise kihina realiseerimine

Kolmandaks leitud lahenduseks on *ois-ws* teenuse kihi realiseerimine uues ÕIS-i projektis eraldi lahtise kihina äriloogika kihi alla nii , et vajadusel suhtleb uue ÕIS-i äriloogika kiht

kolmandate rakenduste ja süsteemidega ning kui ei ole tarvis siis läheb temast mööda. (Joonis 7)



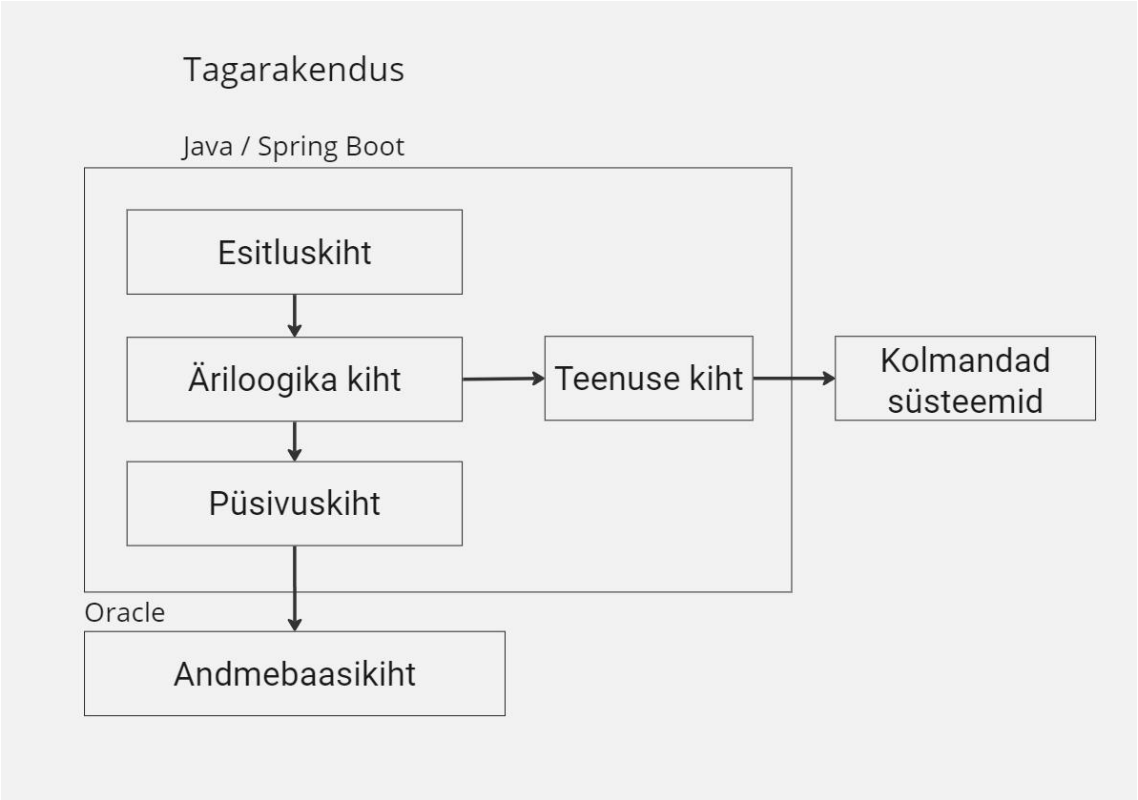

Joonis 7. Lihtsustatud uue ÕIS-i andmevahetuse voo näide kui teenuse kiht realiseeritakse lahtise kihina

See lahendus paigaldab teenuse kihi projekti sisse, mis eemaldab teenuse kihi arenduse käigus haldamise ebamugavused ning teenuse kiht on piisavalt eraldi, et ei teki segadust failide ja loogika eristamisel.

Antud lahendus võeti peaaegu kasutusele, kuid leiti natuke parem lahendus.

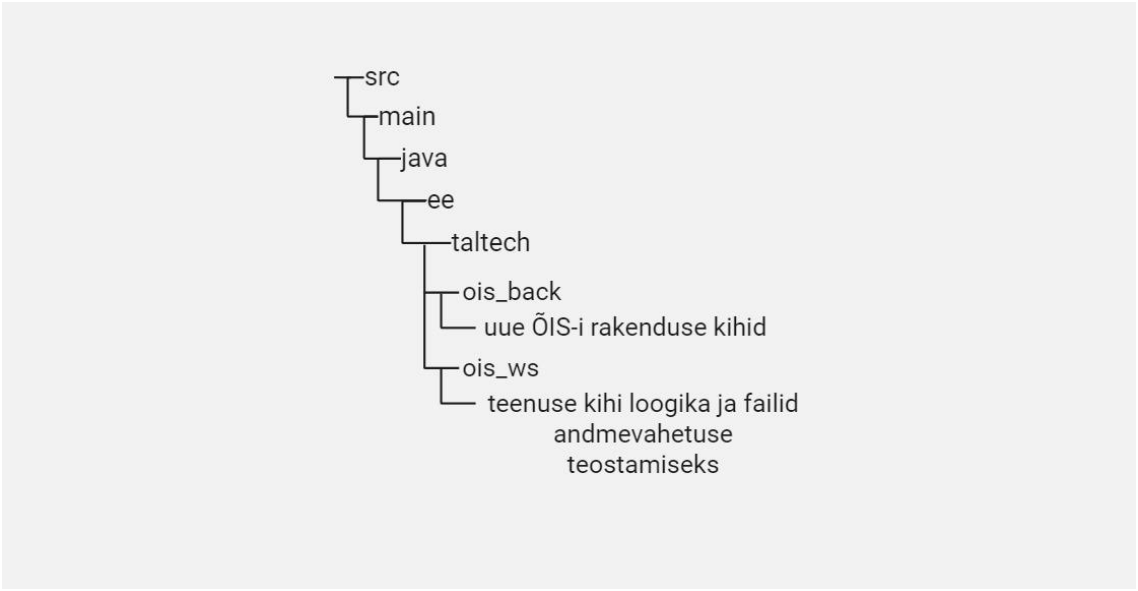
### 3.2.4 Uue ÕIS-i projektis eraldi projekti kihtidest arendamine

Neljandaks ja viimaseks leitud lahenduseks on struktuur, kus *ois-ws* andmevahetuse teenuse kiht paigutatakse ikkagi uue ÕIS-i projekti sisse, kuid olemasolevatest kihtidest eraldi. See lahendus laenab omadusi eelmisest lahendusest. (Joonis 8)




Joonis 8. Lihtsustatud uue ÕIS-i andmevahetuse voo näide kui teenuse kiht realiseeritakse olemasolevatest kihtidest eraldi kihina

Antud stuktruuri lahendus paigutab olemasolevad uue ÕIS-i kihid, mille struktuur ja rollid ei muutu, ja *ois-ws* teenuse kihi loogika ja failid eraldi kasutadesse *ois_back* ja *ois_ws*. (Joonis 9)




Joonis 9. Uue ÕIS-i projekti struktuur kuhu on lisatud teenuse kiht

Kuna antud lahendus, arendades *ois-ws* teenuse kiht uue ÕIS-i projektis olemasolevatest kihtidest eraldi kihina, kasutab omadusi eelnevatest lahendusest, siis on selle lahenduse struktuuri ülesandeks lahendada eelnevate leitud lahenduse miinused, jättes alles võimalikult palju eeliseid ja plusse.

Sellele põhinedes on viimaseks leitud lahenduseks struktuur, kus *ois-ws* teenuse kiht on koos ainukese rakendusega/projektiga, mis teda kasutama hakkab. Lisaks tänu sellele, et *ois-ws* teenuse kiht on paigutatud arendatavasse uue ÕIS-i projekti, pole tarvis teenuse kihi uuendamisel, parandamisel, parendamisel ja arendamisel teha üleliia ebamuguvaid lisa liigutusi. Teenuse kiht on ka piisavalt eraldi, et ei teki segadust *ois-ws* andmevahetus kihi ning  põhiprojekti loogika ja klasside eristamisel.

Lisaks on antud lahenduses *ois-ws* teenuse kihi struktuur paigutatud nii, et kui tulevikus peaks muutuma antud teenuse kiht uue ÕIS-i rakenduse jaoks liiga suureks ja/või seda on plaanis, lisaks uue ÕIS-i projektile mõne uue rakenduse/projekti poolt kasutama hakata, siis teoreetiliselt on võimalik liigutada antud *ois-ws* teenuse kiht mugavalt eraldi projektiks.

### 3.2.5 Tulemus

*Ois-ws* teenuse kihti paigutamise probleemi lahendiks valiti neljas leitud lahendus, kus *ois-ws* teenuse kiht realiseeritakse uue ÕIS-i projekti olemasolevatest kihtidest eraldi kihina. (Joonis 6)

Selle lahenduse valiku põhjusteks on eelmainitud eelised:

- Haldamine – Kuna teenuse kiht on arendatava uue ÕIS-i projektis sees eraldi kihina, on arendajatel, eriti arendamise käigus, mugav ja lihtne teha vajalikke uuendusi ja parandusi ilma, et peaks tegema ebamugavaid lisa liigutusi iga muudatuse järel.
- Arusaadav struktuur – Antud teenuse kiht/teenus on piisavalt eraldi, et ei teki segadust *ois-ws* teenuse kihi ja põhiprojekti failide ja loogika eristamisel.
- Suurendamis võimalus – ois-ws on pakitud kokku teenuse kihi/teenuse failid ja loogika, et vajadusel on teoreetiliselt mugav ja lihtne liigutada antud teenuse kiht/teenus eraldi projekti.

## 3.3 Teenuse kihi sisemise struktuuri analüüs

Nüüd kui on valitud *ois-ws* teenuse kihi paigalduse lahendus, on tarvis analüüsida ja realiseerida teenuse kihi sisemine struktuur tulevaste loogika, failide ja klasside paigutamiseks.

Lõpudokumentide EHIS-esse saatmise analüüsist (Vaata punkt 4.2) on teada, et teenuse kiht hakkab kasutama andmevahetuseks *SOAP* protokolli ja lisaks sellele ka *REST API* arhitektuuri. Alguseks lisame siis ois_ws teenuse kihi alla kaks kausta *rest* ja *soap,* kuhu paigutatakse vajalik admevahetuse loogika ning *config* kaust, kus vajadusel konfigureeritakse vajalikud klassid. (Joonis 10)

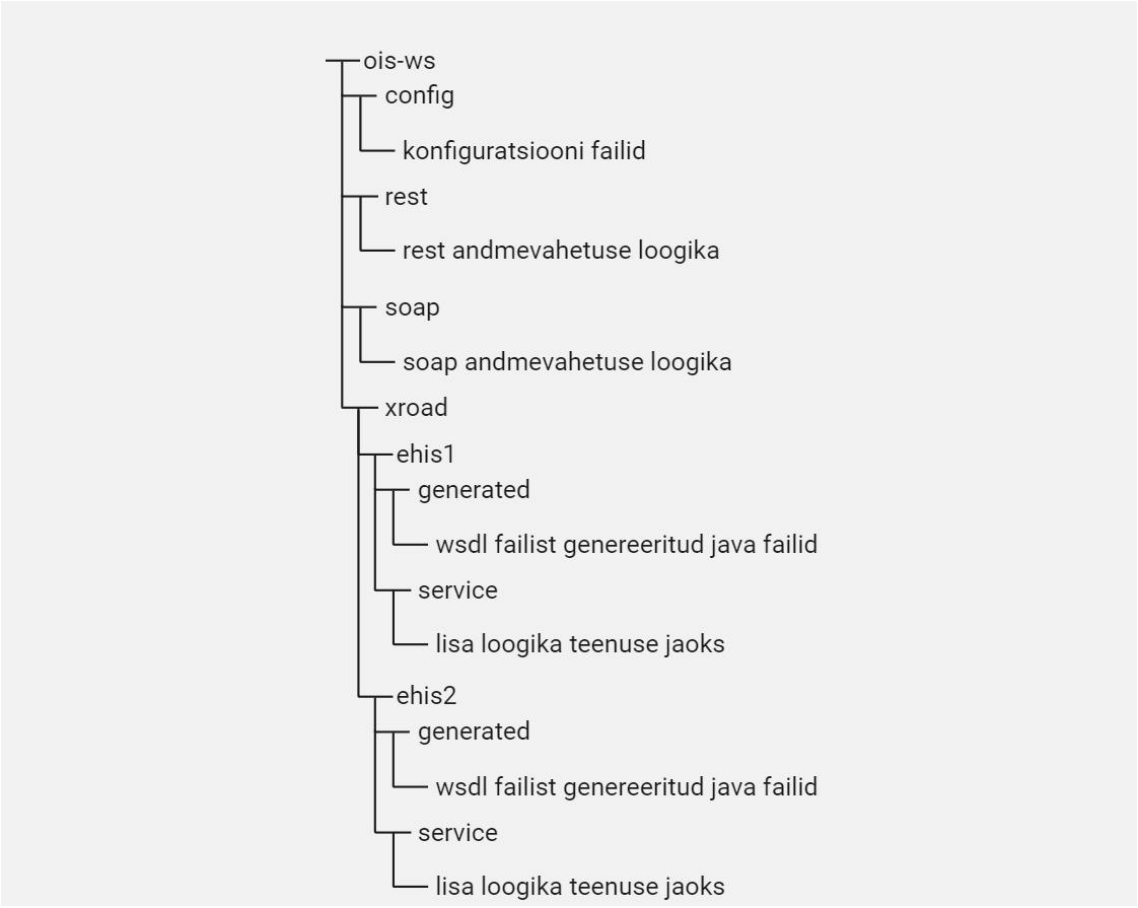

Joonis 10. Teenuse kihi algne struktuur

Lõpudokumentide EHIS-esse saatmise analüüsist (Vaata punkt 4.2) on ka teada, et on tarvis genereerida kahe X-tee [1] teenuse *sendCertificate* ja *adsServce* java failid ja klassid.

X-tee [1] lõudokumentide saatmiseks vajalike teenuste genereeritud failid paigutatakse *ois_ws* teenuse kihi sees olevatesse *xroad/ehis1/generated* ja *xroad/ehis2/generated* kasutadesse. Lisaks on mõlemas *xroad/ehis1* ja *xroad/ehis2* kaustades veel *service* kaustad, kuhu vajadusel paigutatakse andmevahetuseks vajalik lisaloogika. (Joonis 11) (Vaata punkt 4.3.2)

Põhjus, miks *X-tee* [1] teenuste genereeritud klasse, faile ja lisatavat loogikat ei paigutata soap kausta, on teenuste suuruses. Antud teenustel on palju faile ja klasse ning nad on ise suure ja keeruka loogikaga.  Lisaks on sedasi võimalik paremini eristada X-teed [1]

kasutavaid teenuseid ja andmevahetus funktsionaalsus ning lihtsamaid andmevahetuse funktsioonaalsusi.




Joonis 11. Teenuse kihi struktuur, kuhu on lisatud vajalikud genereeritud failid

Teenuse kihi sisemine struktuur on paika pandud ning hetke seisuga ta ei muutu. Kaetud on kõik vajalikud alad, mis on teenuse kihi rollis ja ülesandes: *SOAP*, *REST*, X.-tee [1]. (Joonis 11)

# 4 Lõpudokumentide saatmine EHISesse

Lõpudokumentide EHISesse saatmise analüüsi teostasti koos tiimi analüütikuga. Anaüüsiti *SendCertificate ver 3* [10] *X-tee* [1] teenust, et teada saada millisel kujul peavad dokumendid olema antud teenuse kasutamiseks ning mida tuleb teha, et antud teenust

kasutada.  Lõpliku analüüsi dokumendi, mida autor ka antud bakalaureuse töös kasutab, koostas analüütik.

## 4.1 X-tee

*X-tee* on andmevahetuse platvorm, mis võimaldab turvaliselt asutuste vahel teavet pärida ja vahetada. Teabe vahetamiseks jagab *X-tee* liige oma andmed ning kõik teised liikmed saavad kokkuleppe alusel seda infot kasutada. Kuna *X-tee*-ga liitunud süsteeme on palju, saavad *X-tee* liikmed oma äriprotsesside tõhustamiseks kasutada teiste liikmete loodud e-teenuseid ja andmeid. [1]

## 4.2 Analüüs

Iga lõpetanud tudengi kohta tuleb edastada lõpudokumentidega seotud metaandmed ja PDF-id EHISesse. Lõpudokumentide edastamine toimub taustal, mis tähendab, et peale lõpetamist koostatakse tellimus lõpudokumentide edastamiseks. Antud kohustus on alates 01.01.2022.

### 4.2.1 Õppija tingimused

Lõpudokumentide andmete EHISesse saatmiseks peab õppija uue ÕIS-i projektis vastama kindlatele tingimustele.

Esimeseks tingimuseks on, et ei ole tegemist ühisõppekava õppijatega, kus juhtiv pool ei ole Tallinna Tehnikaülikool.

Teiseks tingimuseks on, et õppija lõpetamine või duplikaadi korraldus oleks vastavas ajavahemikus. Kui leidub duplikaadi käskkiri ja sellega seotud õppijal on samas ajavahemikus ka lõpetamise korraldus, siis valib süsteem andmete saatmiseks duplikaadi korralduse.

Järgmiseks tingimuseks peab olemas olema diplom, mille staatus on *trükitud* või *väljastatud*. Lisaks diplomile peab olemas olema ka eesti keelne akadeemiline õiend, millel sammuti on staatuseks *trükitud* või *väljastatud*.

Viimane tingimus on seotud magistri-, ja doktoriõppega, mille puhul  peab olemas olema ka ingliskeelne akadeemiline õiend, mille staatus on *trükitud* või *väljastatud.*

### 4.2.2 Andmevahetuse lühikirjeldused

Antud punktis on lühidalt kirjeldatud andmevahetuse tingimusi ja võimalusi.

Nagu eelmises punktis (Vaata punkt 4.2.1) on mainitud, ühisõppekava andmed, kus Tallinna Tehnikaülikool ei ole juhtiv pool, EHISesse ei edastata.

Lõpudokumente saadetakse ükshaaval (Üks fail koos vastavate metaandmetega), ning ei edastata järgmist faili enne kui eelmine fail on edukalt edastatud.

EHISest on tulnud tingimus lõpudokumentide saatmise piiramiseks ehk ühte tellimusse ei saa kuuluda rohkem kui 500 sobivat õppurit. Kui leitakse üle 500 sobiva õppuri, siis tuleb tellimusi rohkem koostada.

Dokumente on võimalik saata korduvalt siis, kui on teostatud andmete parandus või duplikaadi väljastamisel otsitakse identifikaatori järgi EHISest varem edastatud andmed ja salvestatakse dokument varasema lõpudokumendi redaktsioonina.

Lisaks on võimalik korduvalt dokumente saata kui muudatuste või duplikaadi edastamisel identifikaatori järgi varem edastatud andmeid ei leita ja saadetavat dokumenti kasitletakse uue lõpudokumendina. Kui varasemalt on laetud sama õppekava, omaniku ja numbriga dokument, siis dokumenti uuesti salvestada ei saa.

Eraldi on võimalus ka saata tagasi võetud dokumente.

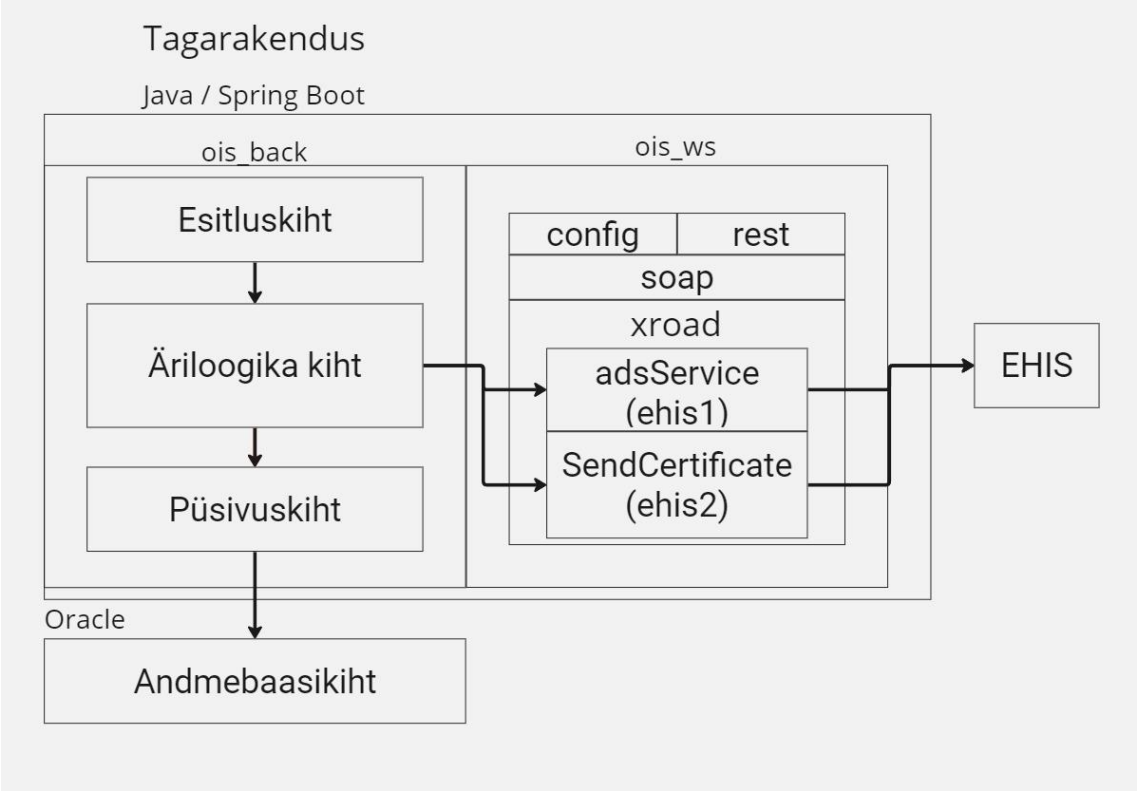
Kui edastatakse EHISe eelselt välja antud lõpudokumendi andmed (õppimise andmed puuduvad), siis lõpetamise andmeid ei salvestata.

Iga leitud õppija andmed saadetakse alati samas järjekorras:

1. Duplikaat – Kui on trükitud diplom märkega *duplikaat* ja trükitud eesti keelne akadeemiline õiend märkega *duplikaat* ning magistri/doktori õppes on õppijal trükitud ka inglise keelne akadeemiline õiend või on trükitud akadeemilise õiendi *duplikaat*.
2. Eksmatrikuleerimine – Kui tegemist on eksterniga, siis saadetakse õppija eksmatrikuleerimise andmed.
3. Juhendamine – Kindlate õppetasemetega õppijad, kellel on olemas kinnitatud lõputöö teema, siis saadetakse juhendamise andmed.

4. Diplom
5. Akad. Õiend e.k.
6. Akad. Õiend i.k.

Andmete edastamiseks võtame kasutusele kaks X-tee [1] teenust *adsService* (ehis1) ja *SendCertificate* (ehis2). (Joonis 12)




Joonis 12. Lihtustatud uue ÕIS-i lõpudokumentide edastamise andmevoo kasutades X-tee teenuseid adsService ja SendCertificate

*AdsService* teenuse kaudu saadetakse lõpudokumentide lisasid: duplikaadi, eksmatrikuleerimise ja juhendamise andmed ning *SendCertificate* teenuse kaudu saadetakse baas lõpudokumendid ja nende andmed: Diplom ja akadeemilised õiendid.

## 4.3 Arendus

Kasutades koostatud analüüsi, arendatakse algul kasutajaliideses (esirakenduses) keskkond, mis võimaldab lõpudokumentide edastamist EHISesse. (Vaata punkt 4.3.1)

Kui esirakenduses on keskkond olemas, hakatakse arendama tagarakenduses vajalikku loogikat esirakenduse erinevate tegevuste jaoks.

Lõpudokumentide saatmise funktsionaalsuseks tuleb vajalikud X-tee [1] teenuste *adsService* ja *SendCertificate* WSDL failidest genereerida vajalikud java failid ning paigutada genereeritud failid realiseeritud *ois-ws* teenuse kihti.

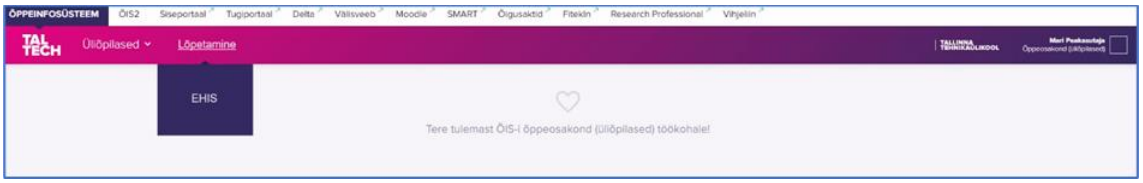
Siis arendatakse antud andmevahetuse jaoks vajalik loogika.

Esitatud pildid on illustreerivad.

### 4.3.1 Esirakendus

Uue dekanaadi töötajatele mõeldud õppeinfosüsteemi esirakenduses on tarvis arendada keskkond, kus kasutajal on võimalik otsida, koostada, vaadata ning saata digitaalseid lõpudokumente EHIS-esse. Selleks on koostatud kliendiga kooskõlastatud analüüs, milline kasutajaliides välja peab nägema.

Kui *Õppeosakond (üliõpilased)* töökohal liikuda menüüle *Lõpetamine – EHIS* (Joonis 13), siis avaneb otsinguvorm, kus kuvatakse otsinguparameetrid varasemate saatmiste kirjete otsimiseks ja vaatamiseks ning nupud *Telli lõpudokumentide saatmine EHISesse*, mis on uute tellimuste tegemiseks, ja *Käivita saatmine* , mis käivitab andmete protessi edastamise. (Joonis 14)




Joonis 13. Õppeosakond (Üliõpilased) töökoha menüü hetke seisuga

Otsinguvormil on järgmised otsinguparameetrid:

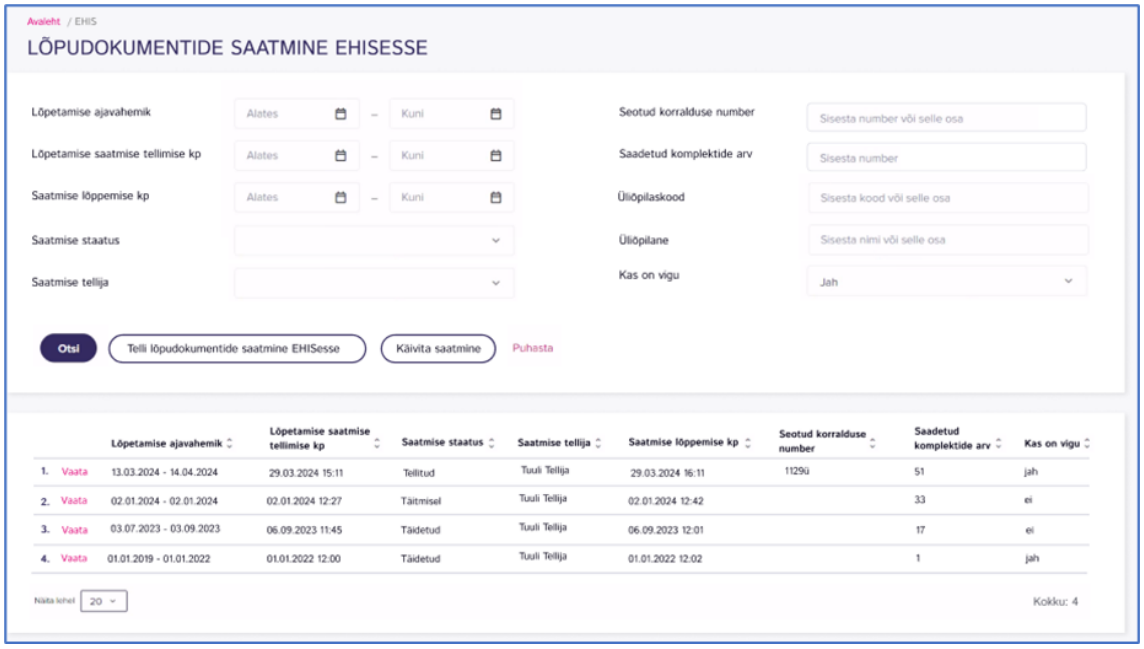
- Lõpetamise ajavahemik
- Lõpetamise saatmise tellimise kuupäev
- Saatmise lõppemise kuupäev
- Saatmise staatus
- Saatmise tellija

- Seotud korralduse number
- Saadetud komplektide arv
- Üliõpilaskood
- Üliõpilane
- Kas on vigu

Vajutades nuppu *Otsi* kuvatakse vastavalt parameetritele sobivad tulemused tellitud saatmiste kohta.

Otsingu tulemuste tabelis kuvatakse järgmised veerud (Joonis 14):

- Rea number
- Nupp *Vaata* – avab tellimuse vaatamise vormi (Joonis 16)
- Lõpetamise ajavahemik
- Lõpetamise saatmise tellimise kuupäev
- Saatmise staatus
- Saatmise tellija
- Saatmise lõppemise kuupäev
- Seotud korralduse number
- Saadetud komplektide arv
- Kas on vigu




Joonis 14. Lõpudokumentide saatmise EHISesse otsinguvorm ja tulemused

Vajutades nupule *Telli lõpudokumentide saatmine EHISesse* avandeb vorm, kus on võimalik valida milliste parameetrite järgi saab andmeid saata. (Joonis 15)

Antud parameetrid on:

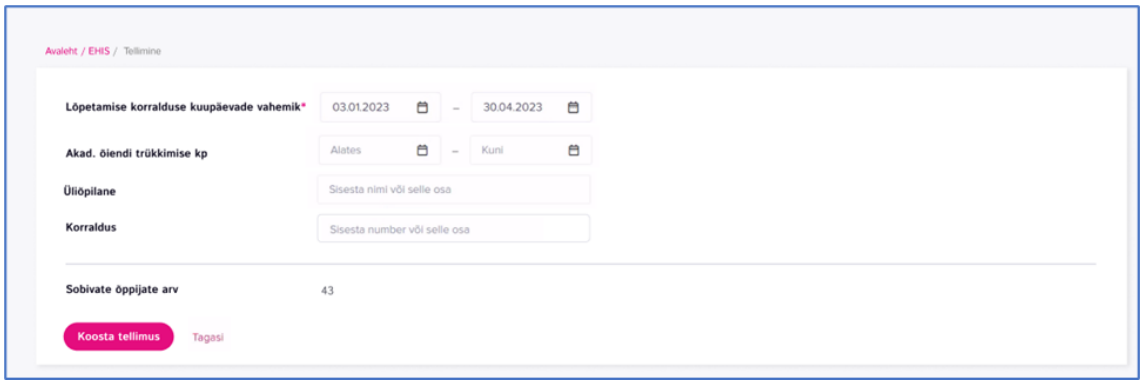
- **Lõpetamise korralduse kuupäevade vahemik** – kohustuslik kuupäevade vahemiku valimise väli. Kui valitud kuupäevade vahemikuga kattub teine lõpudokumentide saatmine EHISesse, mille staatus on *Tellitud* või *Täitmisel*, siis kuvatakse informatiivne teade „Tellimus on juba olemas ajavahemikus: pp.kk.aaaa – pp.kk.aaaa“ antud ajavahemikuga.
- **Akadeemilise õiendi trükkimise kuupäev** – kuupäevade vahemiku valimise väli
- **Üliõpilane** – valiku väli, kus on võimalik otsida sobivate õppijate (õppijad, kes leitakse ette antud parameetritele, lisaks kontrollitakse, et oleks trükitud diplom ja eesti keelne akadeemiline õiend ning inseneri, magistri ja doktorandi tasemel ingliskeelne akadeemiline õiend)
- **Korraldus –** valikuväli, kus on võimalik valida üks korraldus korraga.

Lisaks on näha sobivate õppijate arvu, mis saadakse kui süsteem kontrollib valitud parameetritega, kas leidub tingimustele vastavaid õppijaid ning tagastab saadud õppijate arvu.

Kui vajutada nuppu *Koosta tellimus*, siis koostatakse andmete saatmise tellimus.

Kui tellimuse loomine õnnestus, siis kuvatakse teade „X õpilase lõpudokumendid lisati EHISesse saatmise järjekorda. Andmed saadetakse hajutatult, üldises järjekorras.“.

Kui tellimuse loomine ebaõnnestus, siis kuvatakse vastav vea teade. Näiteks, et ei leitud ühtegi sobivat õppijat.




Joonis 15. Lõpudokumentide tellimuse koostamise vorm

Lõpudokumentide otinguvormil tulemustel *Vaata* nupule vajutamisel avaneb tellimuse detailsed andmed ja saatmise tulemus. (Joonis 16)

Tellimuse vaatamise vormil on näha tellimuse:

- Lõpetamise korraldus alates kuupäev
- Lõpetamise korraldus kuni kuupäev
- Akadeemilise õendi trükkimise algus kuupäev
- Akadeemilise õiendi trükkimise lõpp kuupäev
- Saatmise tellija
- Saatmise staatus
- Saatmise tellimise kuupäev
- Saatmise alustamise kuupäev
- Saatmise lõpetamise kuupäev

Akadeemilise õiendi trükkimise kuupäevad kuvatakse siis, kui need on tellimisel täidetud.

Antud väljade all kuvatakse tabel tudengite andmetega:

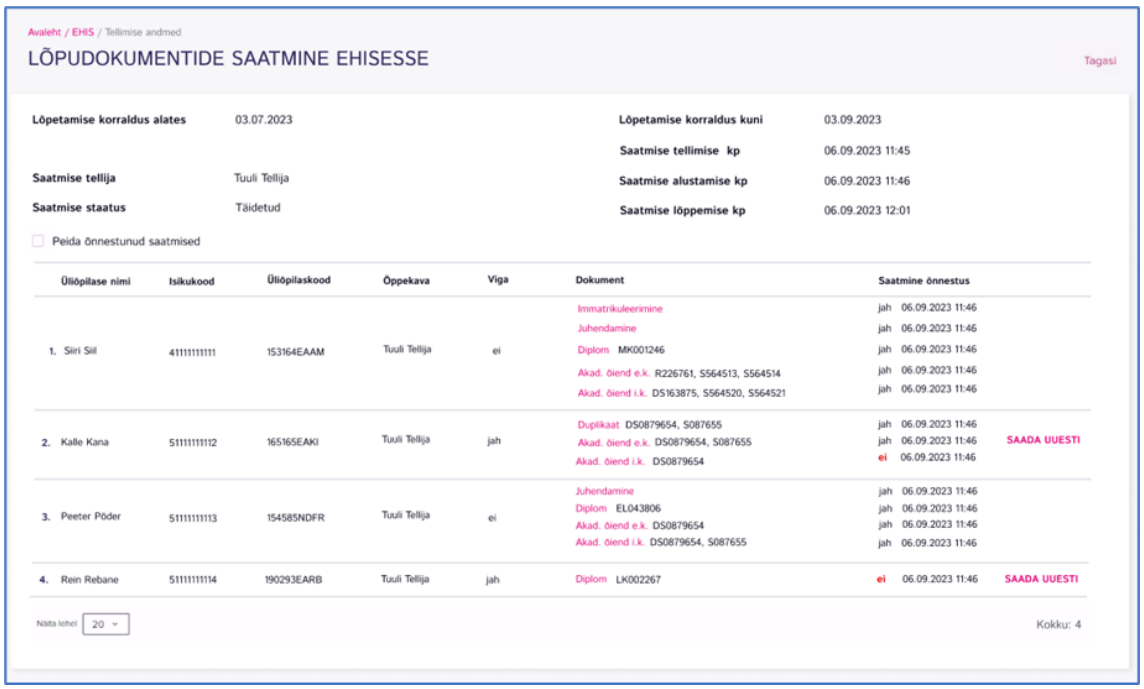
- Üliõpilase nimi
- Isikukood
- Üliõpilaskood
- Õppekava
- Viga
- Dokument – Iga saadetud andmekirje/dokumendi kohta eraldi kirje ja dokumendi(te) number/numbrid.

- Saatmine õnnestus – Õnnestumise väärtus *Jah*/*Ei* ja saatmise kuupäev ning kellaaeg sekundi täpsusega. Kui saatmisega toimus viga, siis kuvatakse nupp *Saada uuesti*, millele vajutades on võimalik komplekt uuesti saata.

Dokument tulbas kuvatavad võimalikud valikud edastamis järjekorras:

7. Immatrikleerimine – Ekstern õppija immatrikuleerimise edastamine EHISesse. Saadetake ainult ekstern õppija korral, kui tegemist on dokumentide esmase saatmisega.
8. Juhendamine – Lõputöö juhendamisega (Lõputöö teema) seotud andmete esitamine EHISesse. Saadetakse ainult magistri ja integreeritud õppe õppijate korral.
9. Diplom – Diplomiga seotud andmete ja faili edastamine EHISesse
10. Akad. õiend e.k. – Eesti keelse akadeemilise õiendiga seotud andmete ja faili edastamine EHISesse
11. Akad. õiend i.k. – inglise keelse akadeemilise õiendiga seotud andmete ja faili edastamine EHISesse.

Lisaks on märkeruut *Peida õnnestunud saatmised*. Kui märkeruut on valitud, siis kuvatakse tudengi andmetega tabelis need kirjed, kus viga on *jah*.




Joonis 16. Lõpudokumentide saatmise tellimuse vaatamise vorm

Lõpudokumentide saatmine EHISesse otsinguvormil vajutades nupule *Käivita saatmine* käivitatakse tellitud kirjete andmete saatmine EHISesse. Kui saatmise käivitamine õnnestus, siis kuvatakse teade „EHISesse andmete laadimine käivitatud“ ning tellimuse staatus muutub *Täitmisel*. Vastasel juhul kuvatakse vea info.

### 4.3.2 Java failide genereerimine

Digitaalsete lõpudokumentide EHIS-esse saatmiseks tuleb kasutusele võtta X-tee teenused *SendCertificate* (ehis2) ja *adsTeenus* (ehis1). Antud teenuste kasutamiseks uue ÕIS-i projektis, tuleb vajalikud Java failid ja klassid genereerida *SendCertificate* ja *adsTeenus* teenuste WSDL failidest, mis on leitavad X-tee kataloogist [11]. Need WSDL on alla laetud ning salvestatud *resources* paketti *wsdl/ehis* kausta, kust nad on lihtsasti leitavad.

Failide genereerimiseks kasutab autor gradle pluginat *wsdl2java* versioon 2.0.2 [12], mis on konfigureeritud *build.gradle* failis. (Joonis 17)

```
Wsdl2java {
	cxfVersion = „4.0.2“
	includesWithOptions = [
		„**/ehis.wsdl“: [’-bareMethods’]
		„**/ehis2.wsdl“: [’-bareMethods’]
	]
}
```

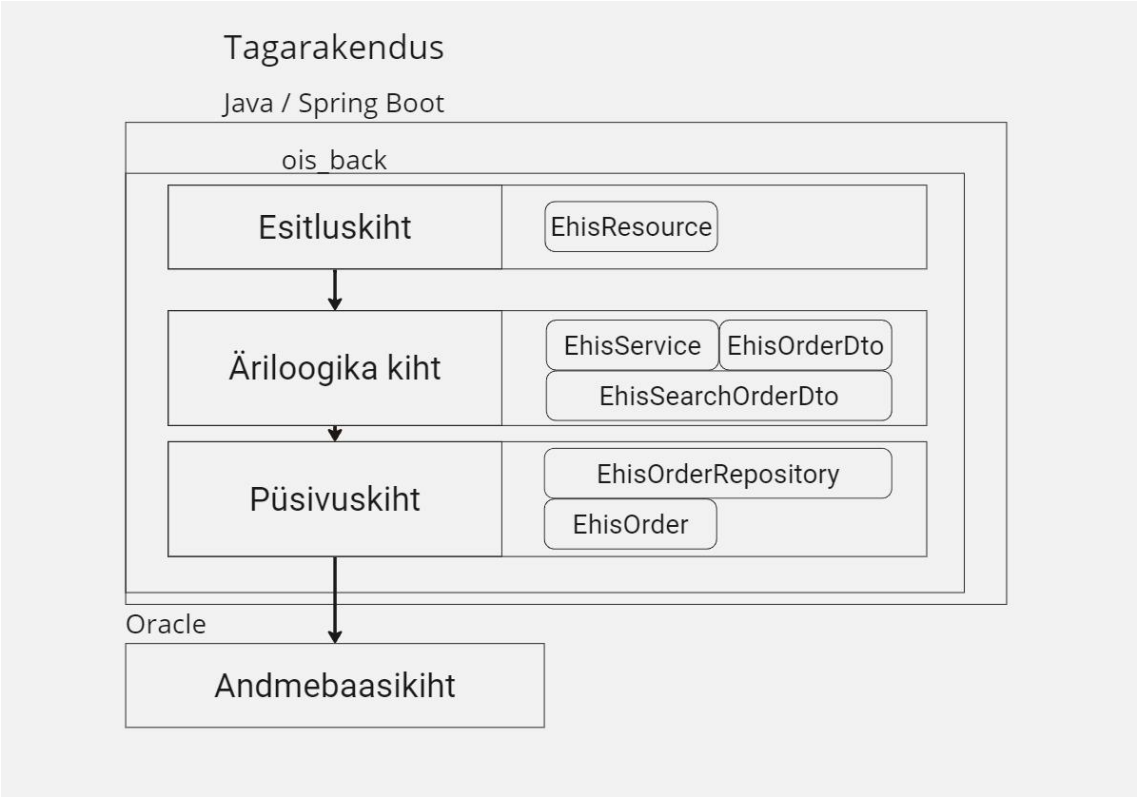
Joonis 17. WSDL failidest Java failide genereerimiseks kasutatud Wsdl2java taski konfigureeringu kood

Failid on genereeritud kasuta *build/generated/wsdl2java/java*, millest tuleb antud failid nende kasutamiseks kopeerida *ois-ws* teenuse kihti antud *xroad/ehis/generated* ja *xroad/ehis2/generated* kautsadesse.

Et ei tekiks probleemi *build* kausta genereetitud failide ja *ois-ws* teenuse kihti kopeeritud failidega, tuleb eemaldada *build* kaustas olevad genereeritud failid.

### 4.3.3 Tagarakendus ja teenuse kiht

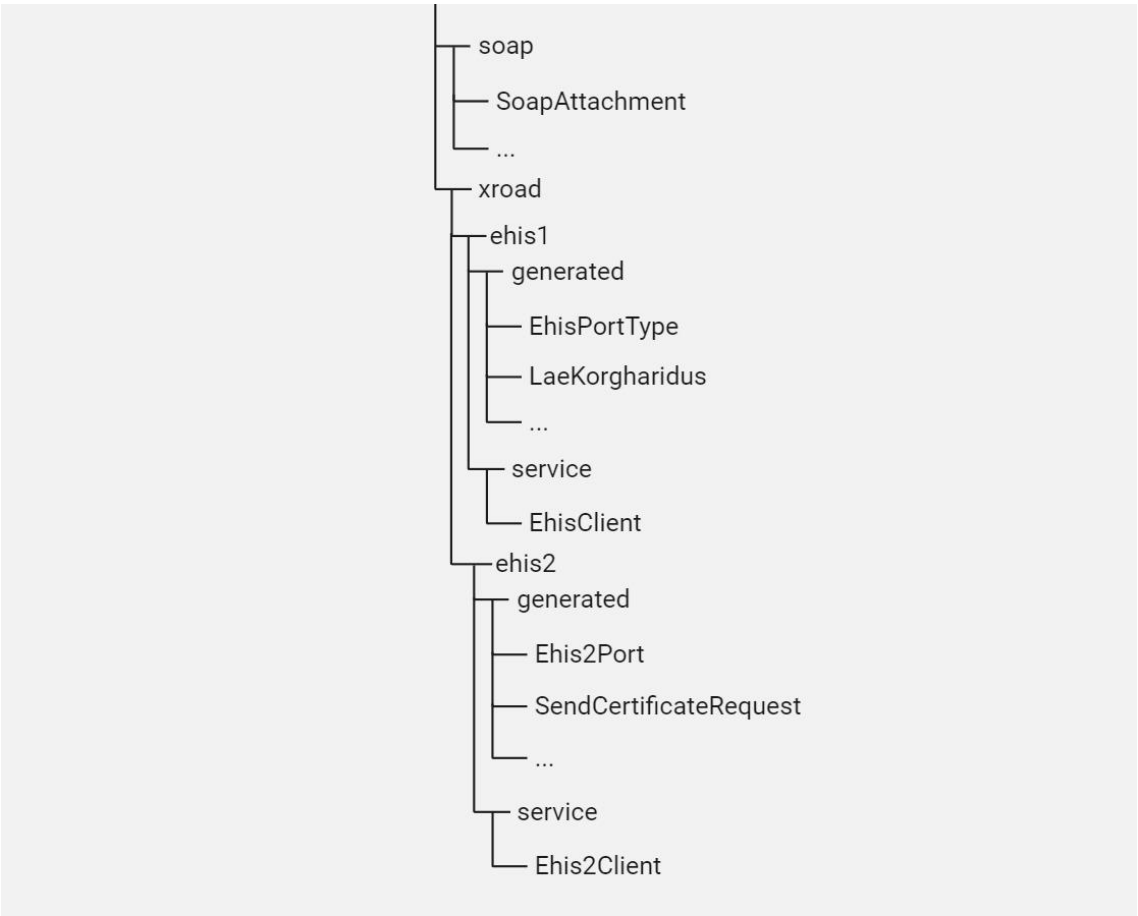
Uue ÕIS-i tagarakenduses on esitluskihis *EhisResource*, äriloogika kihis *EhisService*, *EhisOrderDto* ja *EhisSearchOrderDto* ning püsivuskihis *EhisOrderRepository* ja *EhisOrder* entity klassid, mis suhtevad andmebaasi kihis teatud tabelitega. (Joonis 18)




Joonis 18. EHIS-e klassid uue ÕIS-i projekti kihtides

Antud kihtide ja klasside ülesanne on esirakenduses olenevatele vaadetele ja tegevustele vajaliku funktsionaalsuse tagamine. Näiteks tellimuste otsimine, koostamine ja vaatamine. Lisaks antud tellimuste käivitamine ja uuesti saatmine. (Vaata punkt 4.3.1)

Teenuse kihis on vastavates *ehis/service* ja *ehis2/service* kaustades vastavad *EhisClient* ja *Ehis2Client* klassid. Nendes klassides arendatakse meetodid, mis koostavad lõpliku päringu sisu ning teostavad edastavad antud päringu  EHISesse kasutades genereeritud *EhisPortType* ja *Ehis2Port* klasse. (Joonis 19)




Joonis 19. Teenuse kihi xroad kausta tähtsamad lõpudokumentide edastamise failid ja klassid

Baas dokumentide (diplom ja õiendid) saatmiseks EHISesse tuleb antud dokumenti objektist koostada genereeritud klassi *SendCertificateRequest* objekt. Teiste andmete edastamiseks kasutakse *LaeKorgharidus* klassi objekti. (Joonis 19)

Need objektid koostatakse andmete sisu üleviimisel kasutades *getter*-eid ja *setter*-eid. Antud objektid koostatakse äriloogika kihis *EhisService* klassis, kus igale dokumentile on koostatud eraldi meetod.

Baas dokumentidele lisaks koostatakse *SoapAttachment* klassi object, mis koosneb antud dokumenti *PDF* failist baitide massivina, sisu tüübist (antud juhul *PDF*) ning sisu id-st.

*SoapAttachment* klass on paigutatud *ois-ws* teenuse kihis *soap* kasuta koos logiks vajalike klassidega. (Joonis 19)

Baas dokumentide edastamiseks koostatakse *Ehis2Client* klassi *sendCertificate* meetod, mille sisenditeks on antud *SendCertificateRequest* ja *SoapAttachment* klassid. Lisaks nendele on ka sisendiks *XroadHeader* klass, mis sisaldab antud kooli vajalikke andmeid. Antud meetodis initsialiseeritakse andmevahetuseks vajalik *Ehis2Port* klassi objekt. Sinna objekti lisatakse *XroadHeader* objekti andmed ja *SoapAttachment PDF*. Kui vajalikud andmed on lisatud, siis teostatakse päring kasutades *Ehis2Port* klassi objekti *sendCertificate* meetodit, mille sisendiks on dokumendist koostatud *SendCertificateRequest*.

Vastuseks saadakse *ehis2/service* kaustas oleva *Ehis2SendCertificatesResponse* klassi objekti, mis extendib genereeritud *SendCertificateResponse* klassi. Antud vastus logitakse ning teostatakse järgmised sammud vastavalt vajadusele.

Teiste dokumentide edastamiseks koostatakse *EhisClient* klassi *laeKorgharidus* meetod, kus initsialiseeritakse *EhisPortType* klassi objekt. Antud objekti lisatakse *XroadHeader* objekti andmed. Pärast andmete lisamist, teostatakse päring kasutades *EhisPortType* objekti *laeKorgharidus* meetodit, millse sisendiks on koostatud *LaeKorgharidus* objekt.

Vastuseks saadakse genereeritud LaeKorgharidusResponse objekt, millega teostatakse järgmised sammud vastavalt vajadusele

# 5 Teenuse kihi REST pool

Teenuse kihi REST osa on leitav *ois-ws* teenuse kihi *rest* kaustast, kus on koostatud *RestClient* liides ning kaust iga lihtsama vajaliku *REST API* arhitektuuril põhineva andmevahetuse funktsionaalsuse jaoks.

Liides *RestClient* koosneb teenuse kihis *REST API* kasutavate klasside ühismeetoditest: *checkResponse*, *getHeaders* ja *getRequestEntity*. Iga *REST API*-t kasutav klass implementeeb *RestClient* liidest ning koostab antud implementeeritud meetodid enda andmevahetuseks vajaliku loogika, klasside ja andmetega. Antud *REST API* andmevahetuse klasse ja meetodeid kasutavad siis põhiprojekti äriloogika klassis kindlad *Service* klassid, millega antud andmevahetus seotud on.

*REST* päringu tegemiseks kasutakse Spring raamistiku *RestTemplate* klassi [13], mis on *ois-ws* teenuse kihi *config* kaustas *RestTemplateConfiguration* klassis *Bean*-ina konfigureeritud [14].

*RestTemplate* klassil on mitmeid meetodeid ning *REST* päringu saatmiseks kasutatakse *exchange* meetodit, mis tagastab HTTP staatus koodi ja olenevalt ka päritud andmed.

# 6 Kokkuvõte

Lõputöö eesmärk oli realiseerida uude dekanaadi töötajatele mõeldud õppeinfosüsteemi andmevahetuse teenuse kiht, läbi mille on võimalik nii *SOAP* protokolli kui ka *REST API* arhitektuuri kasutades suhelda kolmandate rakenduste ja süsteemidega.

Antud bakalaureuse töö raames analüüsiti erinevaid teenuse kihi struktuuri võimalusi, analüüsiti koos tiimi analüütikuga lõpudokumentide saatmist EHISesse ning realiseeriti parim leitud teenuse kiht lõpudokumentide EHISesse saatmise näitel.

Kahjuks teema vahetusega tõttu muutus lõputöö tegemine kiireks ja keeruliseks, kuid autor suutis leida ja analüüsida erinevaid lahendusi teenuse kihi realiseerimiseks ning tänu tiimi analüüliku abile ka analüüsida lõpudokumentide saatmist EHISesse.

Kuna lõpudokumentide EHISesse edastamise funktsionaalsus on suur osa ning kogu selle funktsionaalsuse realiseerimine võtab kaua aega, jõuti ka realiseerida *ois-ws* teenuse kihti *REST API* arhitektuuril andmevahetuse funktsionaalsused.

Valitud Teenuse kihi struktuur on tõestanud end sobivaks. Teenuse kihti on vajadusel lihtne ja mugav hallata ka mitme arendaja poolt. Samaaegne andmevahetuste funktsionaalsuste arendamine teenuse kihti ilma suurtemate vigade ja probleemiteta valideeris seda.

EHISega andmevahetuse valideerimiseks testitakse antud funktsionaalsust firma siseselt testija poolt kui ka kliendi enda poolt.

# Kasutatud kirjandus


[1] „X-tee“ [Võrgumaterjal]. Available: Available: https://www.x-tee.ee/home. [kasutatud 04 05 2024]

[2] „React“ [Võrgumaterjal]. Available: https://react.dev. [Kasutatud 22 04 2024]

[3] „TypeScript“ [Võrgumaterjal]. Available: https://www.typescriptlang.org. [Kasutatud 22 04 2024]

[4] „React Bootstrap“ [Võrgumaterjal]. Available: https://react-bootstrap.netlify.app. [Kasutatud 22 04 2024]

[5] „Java“ [Võrgumaterjal]. Available: https://www.java.com/en/. [Kasutatud 22 04 2024]’

[6] „Spring Boot“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-boot#overview [Kasutatud 22 04 2024]

[7] „Gradle Build Tool“ [Võrgumaterjal]. Available: https://docs.gradle.org/current/userguide/userguide.html. [Kasutatud 22 04 2024]

[8] Dean Rubin „The Three Layered Architecture“ [Võrgumaterjal]. Available: https://medium.com/@deanrubin/the-three-layered-architecture-fe30cb0e4a6. [Kasutatud 24 04 2024]

[9] Mark Richards „Software Architecture Patterns“ [Võrgumaterjal]. Available: https://www.oreilly.com/content/software-architecture-patterns/. [kasutatud 01 05 2024]

[10] Kairit Peekman „SendCertificate ver 3 (lõpudokumendi salvestamine)“ [Võrgumaterjal]. Available: https://projektid.edu.ee/pages/viewpage.action?pageId=227352133. [Kasutatud 21 04 2024]

[11] „X-tee alamsüsteemide kataloog teenuste ja WSDL kirjeldustega“ [Võrgumaterjal]. Available: https://x-tee.ee/catalogue/EE. [Kasutatud 21 04 2024]

[12] Bjørn Mølgård Vester „wsdl2java-gradle-plugin“ [Võrgumaterjal]. Available: https://github.com/bjornvester/wsdl2java-gradle-plugin [Kasutatud 21 04 2024]

[13] „Class RestTemplate“ [Võrgumaterjal]. Available: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html. [kasutatud 09 05 2024]

[14] „Basic Concepts: @Bean and @Configuration“ [Võrgumaterjal] . Available: https://docs.spring.io/spring-framework/reference/core/beans/java/basic-concepts.html. [kasutatud 09 05 2024]

[15] „HTML basics“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. [kasutatud 13 05 2024]

[16] „CSS: Cascading Style Sheets“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. [kasutatud 13 05 2024]

[17] Fei Peng “Basic Architecture of Spring Boot-based Rest API” [Võrgumaterjal] Available: https://medium.com/@epengfei/basic-architecture-of-spring-boot-based-rest-api-d0c1c5f128f8. [kasutatud 21 05 2024]

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

Mina, Robert Samuel Roos

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Teenuse kihi realiseerimine uude dekanaadi töötajate õppeinfosüsteemi EHISe digitaalsete lõpudokumentide saatmise näol“, mille juhendaja on Evelin Halling
   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.

09.05.2024

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.