TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Maiko Metsalu IADB185064

Universaalne veebipõhine platvorm põhikooliõpilaste mänguliseks e-õppeks

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

Autorideklaratsioon

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

Autor: Maiko Metsalu

19.04.2023

Annotatsioon

Antud bakalaureusetöö eesmärk on luua Eesti põhikoolide õpilastele ning õpetajatele veebiplatvorm, mis toetab nende õppeprotsessi ning lahendab olemasolevate veebiplatvormide puudujääke.

Töö analüüsi käigus selgitatakse käsitletud probleemi, tuuakse välja olemasolevate platvormide puudused ning luuakse ülevaade arendusprotsessis kasutatavatest tööriistadest.

Arendusprotsessi käigus valmib veebiplatvorm, mis võimaldab õpetajatel ning õpilastel registreeruda ning sisse logida. Õpetajatel on võimalik luua oma küsimuste pank, kuhu on salvestatud nende küsimustikud, ning oma loodud küsimustikega luua oma klassi õpilastele mälukaardi tüüpi mänge. Õpilastel on võimalik loodud mänge mängida ning läbi selle koguda oma profiilile tasemeid.

Antud bakalaureusetöö käigus valminud veebiplatvorm koosneb kolmest komponendist, klientrakendus, serverirakendus ning andmebaas. Sealjuures on kõik komponendid vabavaralised ning avalikult kättesaadavad GitHub veebikeskkonnas.

Lõputöös on käsitletud ka fookusgrupiuuringuid Tamsalu Gümnaasiumis, nii analüüsietapis nõuete kogumisel kui ka veebiplatvormi testimise ning tagasisideuuringu etappides.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 69 leheküljel, 7 peatükki, 19 joonist, 1 tabel.

Abstract

Universal Web-based Platform of Playful e-Learning for Primary School Students

This thesis aims to develop a web-based e-learning platform tailored for primary school students and teachers in Estonia, addressing prevalent issues with existing platforms. It commences with a critical analysis of the current e-learning platforms, pinpointing their limitations and providing an overview of the developmental tools employed in this project.

The centerpiece of this thesis is the creation of a new web-based platform. The platform permits registration and login features for students and teachers. It allows teachers to construct a personalized question bank, storing their created question sets. Using these question sets, teachers can create memory card games, which not only engage the students but also allow them to earn levels for their profiles through gameplay.

The e-learning platform is comprised of three main components: a front-end application, a back-end application, and a database. Furthermore, these components are freeware, enhancing accessibility by making them publicly available on the GitHub platform.

This study incorporates focus group research conducted at Tamsalu Gymnasium, with results influencing the analysis and informing the feedback survey sections of this thesis.

The thesis is in Estonian and contains 69 pages of text, 7 chapters, 19 figures, 1 table.

Lühendite ja mõistete sõnastik

AOP	*Aspect-Oriented Programming*, arenduse üks lähenemisviis
API	*Application Programming Interface,* rakendustarkvara liides
CSS	*Cascading Style Sheets,* küljendamisel kasutatav märgistuskeel
DELETE	Andmete kustutamise päringutüüp REST lähenemisviisil
DTO	*Data Transfer Object*, andmeedastusobjekt
GET	Andmete saamise päringutüüp REST lähenemisviisil
Git	Tarkvaraarenduse versioonihaldussüsteem
gRPC	*gRPC Remote Procedure Calls*, Google poolt arendatav tarkvaraprotokoll, alternatiiv REST-ile
HTML	*Hypertext Markup Language*, hüperteksti märgistuskeel
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll
IDE	*Integrated Development Environment*, integreeritud arenduskeskkond
JAR	*Java Archive*, Java kompileeritud koodi arhiiv
JPA	*Jakarta Persistence API,* Java koodis kasutatav spetsifikatsioon
MVP	*Minimum Viable Product,* minimaalne töötav toode
ORM	*Object-Relational Mapping,* objektide konverteerimine
PATCH	Andmete parandamise päringutüüp REST lähenemisviisil
POST	Andmete loomise päringutüüp REST lähenemisviisil
PUT	Andmete uuendamise päringutüüp REST lähenemisviisil
REST	*Representational State Transfer,* arhitektuuriline lähenemine tarkvarale, mis seab kindlad piirid veebirakenduste arendamisele
SQL	*Structured Query Language,* struktureeritud päringukeel
WWW	*World Wide Web*, ülemaailmne veeb

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Internet, nutiseadmed ning infosüsteemid arenevad kiiresti ning Eesti koolide õpetajad on näidanud, et nad suudavad kiiresti muutuvas infotehnoloogiakeskkonnas adapteeruda, võttes kasutusele uusi tööriistu ning veebiplatvorme, mis hõlbustavad nende tööd ning toovad tavapärasesse klassitundi vaheldust. Veebiplatvormid nagu Nutisport, Kahoot!, Quizziz on vaid mõned näited internetis leiduvatest õppeprotsessi toetavatest tööriistadest, kus õpetajad saavad oma õpilastel lasta lahendada õppekava temaatilisi ülesandeid. Mitmed platvormid on põhjendatult võtnud õppetöö toetamiseks lähenemise, kus õpilastele loodud ülesannete lahendamisele kaasatakse mängulisi elemente nagu näiteks edetabelid, punktiskoorid ja tasemed. Mängulised elemendid teevad õpilase jaoks õppeprotsessi interaktiivsemaks, visuaalselt paremini arusaadavaks ning annavad õpilastele motivatsiooni.

Autori arvamusel on kõige olulisem keskenduda õppeprotsessi mängulisemaks tegemisel veebiplatvormide abil just põhikooliõpilastele. Viimane arvamus toetub autori varasematele kogemustele, mis on näidanud, et Eesti algklasside õpetajad ei poolda oma õpilaste tihedat internetikasutust. Gümnaasiumiõpilaste jaoks ei ole aga mängulisus enam prioriteet, kuna selles kooliastmes on õpilastel oluline keskenduda eelolevateks eksamiteks ning tulevikuplaanide tegemiseks. Viimased kaks põhjust ei välista nende kooliastmete õpilaste ja õpetajate kaasamist mängulistesse veebiplatvormidesse vaid on pigem suunavateks teguriteks veebiplatvormide arendajatele, kes saavad antud asjaolule tuginedes teha disainiotsuseid ning muid platvormi arendusega seotud valikuid.

1.1 Probleem

Eesti põhikoolide õpilaste ja õpetajate poolt kasutatavate veebiplatvormide ja nende kasutamise kohta on välja toodud mitmeid probleeme. Õpetajate populaarsemateks mureallikateks on platvormide kõrge hind ning piiratud võimalused tasuta versioonides, ühekülgne ning korduv protsess ning platvormide õppekeerukus. Viimased kaks on omavahel ka seotud, kuna ühekülgsed veebiplatvormid motiveerivad õpetajaid leidma ja

kasutusele võtma uusi platvorme, mille tõttu õppeprotsess algab otsast peale. Õpilased on välja toonud, et mängulisi veebiplatvorme kasutatakse õppeprotsessis pigem ajaviiteks või vahelduseks, mitte toetavate tööriistadena.

Käesolevas diplomitöös keskendub autor eelnevalt kirjeldatud probleemidele: veebiplatvormide hind, ühekülgne protsess, veebiplatvormide õppekeerukus ning nende sobivus õppeprotsessi toetava tööriistana.

1.2 Eesmärk

Käesoleva töö autori eesmärk on kaardistada Eesti põhikoolide õpilaste ja õpetajate murekohad nende poolt hetkel kasutatavate mängulist õpet toetavate veebiplatvormidega ning analüüsides nende nõudeid ja soove, luua uus veebiplatvorm, mis lahendab praegused murekohad. Antud diplomitöö eesmärk põhineb autori eesmärgil, kuid keskendub vaid autori poolt välja selgitatud kõige olulisematele probleemidele ja nõudmistele. Viimasest asjaolust tulenevalt on eesmärk luua avatud lähtekoodiga vabavaraline veebiplatvorm, mis loob Eesti kooliõpilastele ja -õpetajatele lihtsa ning intuitiivse viisi luua oma õpilastele ülesandeid. Loodavaid ülesandeid saab lahendada mängulised, kuid samas õpilaste teadmisi arendavad.

Loodav veebiplatvorm on diplomitöö piiratud mahu tõttu minimaalne töötav lahendus.

1.3 Ülevaade

Käesolevas töös välja toodud probleemide lahendamiseks ja uue veebiplatvormi loomiseks mis täidab töös seatud eesmärke, on autor otsustanud kõigepealt luua ülevaate lahendatavast probleemist. Samuti on autor analüüsinud eksisteerivaid platvorme ning nende kasutamisega esinevaid probleeme. Eelneva põhjal on autor loonud ülevaate protsessist, millest töö käigus lähtutakse. Nendele etappidele järgneb põhjalik analüüs loodava veebirakenduse nõuetele ning kasutatavatele võetavate tehnoloogiate valikule. Järgnevalt on välja toodud analüüsi käigus valitud tehnoloogiate abil veebiplatvormi arendusprotsess, mille tulemuseks on planeeritav minimaalne töötav lahendus, mis täidab võimalikult hästi diplomitöö eesmärki. Viimaseks on välja toodud valminud lahendust kasutatavate Eesti koolide põhikooliõpilaste ning -õpetajate tagasisideuuringu tulemused ning autori tulevikuplaanid valminud veebiplatvormiga.

2 Metoodika

Käesolevas peatükis on välja toodud käesolevas diplomitöös käsitletud probleemi ülevaade ning välja on toodud kolm eksisteerivat platvormi.

Probleemi käsitlemiseks viis töö autor läbi fookusgrupi küsitluse Tamsalu Gümnaasiumi põhikooliõpilaste ja -õpetajate seas. Küsitlus viidi läbi Google Vormid keskkonnas.

Küsitluste vastuste põhjal tegi töö autor vastustest kasutajalood [7], mida on täpsemalt kirjeldatud töö kolmandas peatükis.

2.1 Ülevaade probleemist

Kooliõpilaste õppimist ja õpetamist toetavad veebipõhised platvormid on internetis laialt levinud ning mõlemale osapoolele kergesti kättesaadavad. Autori arvates on selliste platvormide üldine eesmärk õppeprotsess teha interaktiivseks, lisades õppeprotsessile juurde mängulisi ja võistluslikke elemente, saavutades nii õpilaste parema keskendumise, tähelepanu ja seetõttu ka parema õpiedukuse. Tallinna ülikooli e-õppe keskuse haridustehnoloog Veronika Rogalevitš on koostanud loetelu kolmekümnest erinevast digivahendist, mida õpetajad saavad kasutada abistavate vahenditena õppeprotsessi käigus [1]. Käesolevas töös keskendub autor sellistele platvormidele, mis teevad õppimise õpilaste jaoks mängulisemaks, s.t platvormid, kus õpilased saavad lahendada õppeprogrammis käsitletavaid teemade ülesandeid selliselt, et nende lahendamisel puutuvad õpilased kokku mänguliste elementidega.

Mängudel põhinev õpe on laialdaselt levinud lähenemisviis, mida kasutatakse õpilaste varajastes õppeprotsessides ning selle efektiivsust toetavad mitmed uuringud [2]. Selle põhjal võib järeldada, et mänguliste elementide integreerimine põhikooliõpilaste õppeprotsessi on kasulik meetod, et õpilaste edukust tõsta. Mänguliste elementide integreerimine õppeprotsessi saab toimuda ka ilma veebipõhiste lahendusteta, kuid antud töö käigus käsitletakse vaid veebipõhiseid lahendusi.

2.2 Eksisteerivad platvormid

Õpetajate poolt kasutatavaid eksisteerivaid platvorme on mitmeid. Tamsalu Gümnaasiumi põhikooliõpilaste õpetajatele suunatud küsitluse vastuste analüüsimise käigus selgus, et kasutatakse järgnevaid keskkondasid: JeopardyLabs, Kahoot!, Quizziz, Padlet, Plickers, Nutisport, Matiffic, English Grammar Online, Write&Improve, Classtime, 5dmudelid, vara.ekoolikott, sisuloome. Küsimustikus esinenud küsimuse „Milliste probleemidega olete kokku puutunud kasutades veebilehti, kus õpilased lahendavad teie loodud ülesandeid?“ vastuste seas oli kaks vastuse tüüpi, mis olid populaarsemad kui kõik teised: veebilehted on tasulised (17-st vastusest 10) ning ülesannete lisamine on keeruline või ülesanded on ühekülgsed (17-st vastusest 13).

Järgnevalt on autor välja toonud ning analüüsinud kahte küsitluse käigus selgunud õpetajate ning õpilaste poolt kasutatavat veebiplatvormi ning ühte enda valitud programmi, mis ei ole veebiplatvorm vaid installeeritav tarkvara.

2.2.1 Nutisport

Nutisport on 2015. aastal loodud Eesti kooliõpilaste matemaatikaõpet toetav veebipõhine platvorm, kus õpilased saavad oma matemaatikaoskuseid arendada. Nutispordiga on Eestis liitunud 123 kooli / tiimi ja 37 700 kasutajat [3], seega võib öelda, et antud platvorm on Eesti koolide seas laialt levinud. Antud platvormil saavad lahendavad õpilased enda valitud teemalisi matemaatilisi ülesandeid kindla aja jooksul, saades õigesti lahendatud ülesannete eest punkte. Ülesanded genereeritakse automaatselt vastavalt õpilase valitud teemale. Autori arvates täidab antud veebiplatvorm oma rolli õppeprotsessi toetamisel suurepäraselt, tehes matemaatikaülesannete lahendamise põnevamaks ning andes õpilastele motivatsiooni harjutades saavutada paremaid tulemusi ning osaleda võistlustel. Viimane arvamus ei ole objektiivne, vaid põhineb suuresti autori enda kogemustel põhikoolis ja gümnaasiumiastmes, kus autor kasutas seda veebiplatvormi tihti.

Antud veebiplatvorm keskendub matemaatikale ja sellest tulenevalt on selle platvormi puuduseks asjaolu, et see toetab vaid õpilaste matemaatika õppimist. Sarnaseid veebiplatvorme teiste õppeainete toetamiseks Eestis ei leidu. „Suure kooli ühe õpilase kohta on hind vaid 50 - 99 senti aastas“ [4]. Järelikult ei ole sellel platvormil probleemi hinnaga. 2023 aasta 11. aprilli seisuga leidis töö autor Nutispordi veebilehelt 77 erinevat

matemaatilist mängu, mis on jaotatud kuute erinevasse kategooriasse, seega ei ole antud platvormi probleemiks ka ühekülgne ülesannete lahendamisviis.

2.2.2 Kahoot!

Kahoot! on mängudel põhinev veebiplatvorm mis teeb harivate mängude loomise, jagamise ja mängimise kõigi jaoks lihtsaks [4]. Antud platvormi kasutatakse koolitundides õpilaste teadmiste kontrollimiseks, toetades sellega õppeprotsessi. Autori põhikooli- ning gümnaasiumipõhiste kogemuste põhjal kasutavad õpetajad antud platvormi, et luua mänge, mis sisaldavad valikvastustega küsimusi, millele õpilased reaalajas vastavad. Vastamisel loeb ka kiirus, ning iga õige vastuse puhul saavad õpilased küsimuste eest punkte vastavalt sellele, kui kiiresti vastati. Läbi selle saavad õpilased teineteisega konkureerida ning võivad seetõttu saada rohkem motivatsiooni õppeprotsessis rohkem pingutama.

Antud platvormi saavad kasutada kõikide õppeainete õpetajad, sest platvorm ei genereeri küsimusi ise, vaid see roll on jäetud õpetajale. Selle veebiplatvormi puudusteks on aga Eesti keele toe puudumine ning õpetajate võimaluste vähesus. Samuti on antud platvormi tasuta versioon on rangelt limiteeritud. Platvormi kõige odavam versioon õpetajale maksab 17€ kuus või 204€ aastas, ning antud paketil on range limiit 20 õpilast iga mängu kohta [6].

2.2.3 Alguse Asi II

Alguse Asi II on eestikeelne last hariv arvutiprogramm, mis on mõeldud algklasside õpilastele [5]. Antud platvorm on alternatiiv veebipõhisele platvormile, kuna tegemist on programmiga mis tuleb arvutisse installeerida. Platvormil on võimalik õpilasel harjutada matemaatikat, kirjatähtede kirjutamist, lippude arvamist ja inglise ning saksa keelsete võõrsõnade õppimist. Programmil on kasutajaliides, mis on oma visuaalsete elementidega suunatud just algklasside õpilastele.

Alguse Asi II programmi eeliseks on asjaolu, et see on eestikeelne. Selle kasutamiseks on aga vaja litsentsi. Samuti pole see veebipõhine ning on loodud vananenud tehnoloogiatega. Seetõttu ei pruugi antud programm tänapäeval levinud operatsioonisüsteemides ka kasutatav olla. Kuna tegemist on vananenud tehnoloogiatega, siis ei pidanud töö autor vajalikuks litsentsi hinda ning platvormi funktsionaalsust lähemalt uurida.

2.3 Ülevaade protsessist

Eelnevalt mainitud platvormide puudused saab üldiselt kokku võtta järgnevalt: antud platvormidel ei ole õpetajatel häid võimalusi anda oma sisendit, et õpilaste õppeprotsessi toetada, vaid need platvormid toetuvad ülesannete või mängude automaatsele genereerimisele.

Lähtudes eelnevate platvormide puudustest ning enda motivatsioonist teha põhikooli õpilaste õppeprotsess mängulisemaks pakub antud diplomitöö autor välja uue veebipõhise platvormi, mille valmimisel eelnimetatud puudused leiavad lahenduse ning mille arendusprotsessi on kaasatud mitmed Tamsalu Gümnaasiumi põhikooliõpilaste õpetajad.

Eelnimetatud platvormidega on töö autor kokku puutunud nii põhikoolis kui ka gümnaasiumis ning nende platvormide väljatoodud puuduste likvideerimisel lähtub autor enda kogemustest nii õpilase kui ka õpetaja rollis ning diskussioonidest teiste õpetajatega.

Uue veebipõhise platvormi põhieesmärk ei ole vaid eelnevate platvormide puuduste likvideerimine, vaid Eesti põhikooliõpilastele ning -õpetajatele uudse lahenduse loomine lähtudes nende nõuetele ja soovidele. Antud nõuete ja soovide kogumiseks koostas töö autor mõlemale osapoolele küsitluse kasutades Google Vormid keskkonda.

Veebiplatvormi arenduse käigus keskendub autor asjaolule, et uue veebiplatvormi täielik valmimine ei mahu antud diplomitöö skoopi ja peamiseks eesmärgiks on valmis saada *MVP* ehk minimaalne töötav toode. Õpilaste ja õpetajate nõuete ja soovide prioritiseerimiseks kasutas töö autor MoSCoW tehnikat, mis jagab nõuded ja soovid nelja kategooriasse:

- *Must have* (peab olema)
- *Should have* (peaks olema)
- *Could have* (võiks olla)
- *Won't have this time* (seekord ei pea olema)

Veebiplatvormi arenduse käigus keskenduti vaid „Peab olema“ kategooriale, lähtudes ideest, et ilma sellesse kategooriasse kuuluvate nõuete ja soovide realiseerimiseta ei ole uuel veebipõhisel platvormil eesmärk täidetud [6].

Veebiplatvormi arenduse viimane etapp on valminud veebipõhise platvormi kasutatavuse testimine kahes fookusgrupis mis koosnevad Tamsalu Gümnaasiumi 5. – 9. klasside õpilastest ning õpetajatest ning nende tagasiside kogumine läbi fookusgrupi küsitluse, mis koostati Google Vormid keskkonnas.

Antud veebipõhise platvormi eesmärk ei ole tulu teenida, vaid põhikooli õpilaste ning õpetajate õppeprotsessi toetamiseks uue keskkonna loomine. Sellel põhjusel on töö autor otsustanud kogu projektiga seotud lähtekoodi teha avalikuks, et kõigil õpetajatel, õpilastel ja teistel huvilistel oleks võimalus antud platvormi arendusse panustada. Uue veebiplatvormi majutamiseks arenduse ning testimise käigus kasutab töö autor oma isiklikku serverit, mille kulud katab töö autor ise. Autoril on idee tulevikus teha koostööd Haridusministeeriumiga, et projekti populaarsemaks muutumisel koos leida võimalusi antud veebiplatvorm paremini majutada.

3 Veebirakenduse analüüs

Nõuete määramisel lähtuti ideest, et loodav platvorm on mõeldud Eesti põhikooliõpilastele ja -õpetajatele ning peaks eelkõige lahendama käesolevas töös välja toodud probleemid olemasolevate platvormidega. Veebiplatvorm peab toetama õppeprotsessi tehes seda mängulisemaks, sest mänguliste elementide lisamine õppeprotsessi on näidanud, et õpilaste õppetulemused paranevad [7].

Autor koostas järgnevalt välja toodud nõuded tuginedes küsitluste vastustele ning oma varasematele töökogemustele arendus- ning haridusvaldkonnas. Antud töös on välja toodud nõuded, mille täitmine mahtus töö skoopi. Nõuded, mille täitmine töö skoopi ei mahtunud, on välja toodud alapeatükis „Edasiarendused tulevikus“.

3.1 Nõuded veebirakendusele

Veebirakenduse nõuete kogumiseks viis autor läbi fookusgrupi küsitluse. Kokku vastas küsitlusele 17 Tamsalu Gümnaasiumi õpetajat, kes on praegusel õppeaastal või varasemalt õpetanud põhikooliõpilasi. Õpilaste seas oli küsitlusele vastanuid 103.

Loodud kasutajalood jaotas töö autor kahte rolli:

- Õpilane
- Õpetaja

3.1.1 Mittefunktsionaalsed nõuded

Õpilase rolli mittefunktsionaalsed nõuded on järgnevad.

- Õpilasena soovin, et veebileht oleks eestikeelne
- Õpilasena soovin, et veebilehe kasutamine ei vajaks eraldi õppimist
- Õpilasena soovin kindel olla oma andmete turvalisuses
- Õpilasena soovin, saan küsimustele vastata mänguliselt

Õpetaja rolli mittefunktsionaalsed nõuded on järgnevad.

- Õpetajana soovin, et küsimuste panga loomine ja jagamine oleks intuitiivne
- Õpetajana soovin, et mängude loomine oleks kiire protsess
- Õpetajana soovin, et mängude mängimine tunnis ei peaks nõudma palju aega

3.1.2 Funktsionaalsed nõuded

Õpilase rolli funktsionaalsed nõuded on järgnevad

- Õpilasena soovin registreeruda oma e-maili aadressiga.
- Õpilasena soovin veebiplatvormile sisse ja välja logida
- Õpilasena soovin oma kooli õpetajate poolt loodud mänge mängida
- Õpilasena soovin mängides näha oma progressi, vigu ja õigeid vastuseid
- Õpilasena soovin enne mängimist näha mängu pikkust ja küsimuste arvu
- Õpilasena soovin mänge mängides koguda punkte ja tasemeid
- Õpilasena soovin ennast võrrelda teiste oma klassi- ja koolikaaslastega

Õpetaja rolli funktsionaalsed nõuded on järgnevad

- Õpetajana soovin registreeruda oma e-maili aadressiga
- Õpetajana soovin veebiplatvormile sisse ja välja logida
- Õpetajana soovin luua omale klasse ning määrata nendesse õpilasi
- Õpetajana soovin luua omale küsimustikke
- Õpetajana soovin oma loodud küsimustike põhjal luua õpilastele mänge
- Õpetajana soovin mängu loomisel määrata mängule sihtrühma
- Õpetajana soovin näha iga mängu kohta õpilaste tulemusi

3.1.3 Edasiarendused tulevikus

Antud veebiplatvormile on veel nõudeid, mis käesoleva töö skoopi ei mahu. Siin on välja toodud mõned nõuded, mis on töö autoril tulevikus plaanis realiseerida:

- Õpilasena tahan mängida teiste klassikaaslastega reaalajas
- Õpilasena tahan mängu käigus võrrelda oma progressi teistega
- Õpilasena tahan mängida mänge anonüümselt
- Õpilasena tahan oma kogutud punkte ja tasemeid kulutada oma profiili muutmiseks
- Õpetajana soovin oma küsimuste panka jagada teiste koolide aineõpetajatega
- Õpetajana soovin näha anonüümselt mängitud mängude statistikat
- Õpetajana soovin näha iga õpilase mängude ajalugu ning arengutaseme muutust ajas

Käesolevas töös väljatoodud nõuded ei ole lõplikud ning töö autor on arvestanud asjaoluga, et maailm on muutuv ning nõuded võivad ajas muutuda. Nõuete asjakohasena hoidmiseks jätkas töö autor koostööd töö käigus küsitletud Tamsalu Gümnaasiumi õpetajate ning õpilastega, et nende kõige nõutumad soovid oleksid prioritiseeritud.

3.2 Tehnoloogiate valik

Õigete tehnoloogiate valik on tarkvaraprojekti jaoks ülioluline, sest valitud tehnoloogiatel on otsene mõju loodava süsteemi tõhususele, hallatavusele ja skaleeritavusele [9]. Käesoleva töö autor tugines tehnoloogiate valikul oma varasematele kogemustele bakalaureuseõppes, töökogemusele tarkvaraarendajana, tehnoloogiate populaarsusele ning kasutusmugavusele ja kiirusele [10].

Veebiplatvormi luues lähtus autor HTTP protokollist ning WWW põhimõtetest ja väljapakutud standarditest (RFC 7230 – 7235) [11], mille kohaselt veebiplatvormi arendamisel on mõistlik kasutada REST (*Representational State Transfer*) lähenemist. REST lähenemine tarkvaraarenduses tähendab veebirakenduste vahelise suhtluse

standardiseeritud käsitlemist, kasutades selleks HTTP meetodeid GET, POST, DELETE, PUT, PATCH jt [12].

REST lähenemisviisile on ka alternatiive: GraphQL, gRPC, WebSocket, MQTT jt [13]. Eelnimetatud lähenemisviisidega käesoleva töö autoril kogemused puuduvad ning töö skoopi ei oleks mahtunud uue lähenemisviisiga tutvumine. Alternatiivsete lähenemisviisidega tutvumisel ei leidnud autor selget kasu mõne teise tehnoloogia kasutusele võtmisel.

Järgnevate tehnoloogiate valiku tegemisel oli ka REST lähenemisviisi tugi üheks parameetriks, kuna ilma REST lähenemisviisi toeta oleks veebiplatvormi arendus muutunud autori jaoks keerulisemaks ja ajakulukamaks. Samuti on REST lähenemisviis üks kõige levinumaid standardeid arendusvaldkonnas [15], mis on arendatava veebiplatvormi tuleviku mõttes hea valik, kuna selliseid tehnoloogiaid kasutades on suurem tõenäosus, et leidub ka teisi, kes veebiplatvormi arendusse panustada tahavad.

3.2.1 Tagarakenduse tehnoloogiate valik

Tagarakenduse tehnoloogiate valik peab algama programmeerimiskeele valikuga, kuna erinevate programmeerimiskeelte jaoks on välja töötatud erinevad raamistikud. Järgnevalt on populaarsuse järjestuses välja toodud viis kõige populaarsemat veebiserverite arendamiseks mõeldud programmeerimiskeelt ning iga programmeerimiskeele kohta on autor välja toonud oma arvamuse, miks on see programmeerimiskeel hea valik [14].

- Javascript

Javascripti kasutavad serverirakenduste loomiseks ettevõtted nagu Paypal, LinkedIn, Netflix, Mozilla, Uber jt [15]. Tehnoloogiad, mida kasutavad tuntud suurettevõtted, on kindlasti üks mõjuv põhjust, miks mingit programmeerimiskeelt valida, sest autori arvates teevad sellised ettevõtted eeltööd ning nende valikud on kindlasti põhjendatud.

- Python

Pythoni programmeerimiskeel on populaarsem kui kunagi varem [16]. Python on lihtne programmeerimiskeel, mida on hea õpetada algajatele tänu selle süntaksile, mis on oma

kirjapildilt sarnane inglise keelele. Guido van Rossum, Pythoni programmeerimiskeel looja on öelnud järgnevat: „*Python is designed to be readable and its resemblance to English is an intentional choice, making it easier for developers to understand and maintain the code*“ [17]. Programmeerimiskeele lihtsus teeb veebiserveri arenduse kiiremaks, mis on kindlasti eelis, eriti kui tegemist on ajakriitiliste projektidega.

- Java

Isegi kui sisse arvestada skriptimiskeeled HTML & CSS, siis Java, mis loodi Sun Microsystems'i poolt 1990-ndate alguses [18], oli Stackoverflow Developer Survey 2022 tulemustes populaarsuselt kuues [14]. See näitab, et Java programmeerimiskeel on juba mitu kümnendit olnud arendajate populaarne valik ning seetõttu on see keel järelikult ka hea valik, kui tahetakse valida programmeerimiskeel, mis on küps ning millel on palju kogemustega arendajaid

- C#

C# on modernne objektorienteeritud programmeerimiskeel, mille abil saavad arendajad luua turvalisi ja kindlaid veebiservereid [19]. C# on sarnane Javale, seda programmeerimiskeelt arendab Microsoft [19]. Tänu oma sarnasusele Java keelega on C# hea valik nende jaoks, kes soovivad sarnast turvalisuse taset nagu seda on Java keelel, aga saades kõige modernsemad programmeerimiskeele süntaksi uuendused mis tänapäeval saadaval on. Viimane on kindlasti argument, miks kaaluda C# programmeerimiskeele kasutamist veebiserverite arendamiseks.

- PHP

PHP-d kasutab 79,2% kogu maailma veebilehtedest [20]. Kuigi PHP on üks enimlevinud programmeerimiskeeli üle maailma, ei ole see arendajate poolt väga armastatud keel. PHP kasutuse nii kõrge protsent tuleneb hoopis CMS kasutusest (*Content Management System*). Viimane on tehnoloogia, mis aitab luua veebilehti ilma tehniliste oskusteta [21]. Kuigi antud keel pole arendajate seas väga levinud kui meeldiv programmeerimiskeel [14], on tegemist siiski keelega, mis ei kao lähitulevikus kuhugi, sest enam kui kolmveerand maailma veebisaitidest kasutab seda. See ongi PHP üheks positiivseks omaduseks.

Käesoleva töö autor on kokku puutunud kõigi eelnimetatud keeltega. Javascriptiga on autor loonud kolm veebiserverit, Pythonit õpetab autor aktiivselt oma huviringi õpilastele, Javat kasutab autor oma igapäevatöös, C# ja PHP-ga on töö autor oma õpingute käigus loonud erinevaid veebiservereid. Keeled järjestatuna autori kogemuste põhjal alates keelest millega on autoril kõige rohkem kogemusi:

1. Java
2. Python
3. Javascript
4. C#
5. PHP

Järgnevalt on autor välja toonud eelnevalt välja toodud programmeerimiskeelte populaarsemaid raamistikke, mis on mõeldud veebiserverite arenduseks ning millega autor ise on kokku puutunud. Kindlasti on antud keeltel veel populaarseid raamistikke, kuid käesoleva töö skoopi ei mahu uue raamistiku õppimine, mistõttu ei ole antud raamistikke siin ka välja toodud.

- Java

Spring – Modulaarne Java raamistik, mis sobib kõigi java rakenduste arendamiseks [22]. Sisaldab näiteks järgnevaid raamistikke Spring MVC-d, Spring Core-i, Spring Security’t, Spring Boot-i jt. mooduleid. Spring raamistik on avatud lähtekoodiga.

- Python

Django – Pythoni raamistik, mis toetab kiiret arendusprotsessi ning puhast, pragmaatilist koodistiili [23]. Django, nagu Spring, on ka avatud lähtekoodiga.

- Javascript

Nest.js – Node.js raamistikust edasi arendatud Javascripti raamistik, mis on avatud lähtekoodiga ning keskendub arendaja pandlikkusele ning paljudele levinud koodistiilidele ja standarditele [24].

- C#

.NET – Microsofti poolt loodud ja hallatud avatud lähtekoodiga raamistik, millega saab arendada rakendusi erinevatele platvormidele [19].

- PHP

Phalcon – PHP avatud lähtekoodiga raamistik, mis keskendub arenduskiirusele. „*If frameworks where superheroes, Phalcon would be Flash*“ [25].

Kõik eelnimetatud raamistikud on avatud lähtekoodiga ning omavad lihtsasti kättesaadavat dokumentatsiooni. Järgnevalt on välja toodud tabel programmeerimiskeelte ja nende raamistike võrdluseks, tuues esile programmeerimiskeelte populaarsuse, arendajate poolse meeldivuse tuginedes Stackoverflow Developer Survey 2022 tulemustele ning autori kogemuse hinnangu programmeerimiskeelte ning raamistikega.

Tabel 1. Tagarakenduse keelte ja raamistike võrdlus

Keel	Raamistik	Keele populaarsuse indeks	Keele meeldivuse indeks	Autori kogemus raamistikuga
Java	Spring	3.	4.	8 aastat
Python	Django	2.	1.	1 aasta
Javascript	Nest.js	1.	3.	1.5 aastat
C#	.NET	4.	2.	2 aastat
PHP	Phalcon	5.	5.	1 aasta

Eelnevalt välja toodud autori kogemuste, populaarsuse, autori kommentaaride, viidete ning võrdlustabeli põhjal valis autor tagarakenduse programmeerimiskeeleks Java ning Spring raamistiku. Autor tunneb ennast selle keele ja raamistikuga kõige mugavamalt, ning kuigi nende valikutega ei ole arenduskiirus kindlasti kõige parem, siis tõenäoliselt

kompenseerivad eelnevad kogemused selle aja, mis autoril kuluks teiste raamistike puhul õppimiseks ning probleemide lahendamiseks. Java ning Spring raamistik toetavad väga hästi REST lähenemisviisi, seega on ka see nõue täidetud.

Lähtudes eelnevast valikust tuli autoril teha veel valik kahe populaarse koodiehituse automatiseerimise tööriista vahel: Gradle ja Maven.

Gradle on avatud lähtekoodiga koodiehituse automatiseerimise tööriist, mille eesmärk on automatiseerida java rakenduse loomine [26]. Gradle loomisel peeti silmas selle eelkäiate, Maven'i ja ANT'i puhul esinevaid probleeme, millele üritati lahendus leida [26].

Maven on Apache poolt arendatud tööriist mille eesmärk on sarnaselt Gradle'ile koodiehituse automatiseerimine. See loodi põhimõttega, et Java arendajad saaksid lihtsasti luua JAR faile mida jagada [27].

Mõlemad välja toodud tööriistad on laialdaselt kasutusel ning Spring raamistik toetab neid mõlemaid. Diplomitöö autoril on kogemusi vaid Gradle tööriistaga, mistõttu osutus see valituks.

3.2.2 Klientrakenduse tehnoloogiad

Klientrakenduse tehnoloogiate valiku tegemisel ei ole erinevalt tagarakenduse programmeerimiskeeltega mitut populaarset valikut: 98% kogu maailma veebirakendustest kasutavad Javascripti programmeerimiskeelt [26]. Samuti on see ainuke keel HTML & CSS kõrval, millega käesoleva töö autoril on kogemusi klientrakenduse loomisel. Siinkohal on oluline märkida, et HTML & CSS ei ole programmeerimiskeeled, vaid veebilehtede stiliseerimiseks mõeldud keeled [27].

Klientrakenduse arenduse hõlbustamiseks on Javascripti programmeerimiskeele jaoks välja arendatud mitmeid raamistikke. Järgnevalt on välja toodud kolm raamistikku, millega töö autor on kokku puutunud:

- Angular – Google arendajate poolt loodud ning hallatud populaarne Javascripti raamistik, mis on StackOverflow Developer Survey 2022 andmetel arendajate neljas lemmik raamistik. [14]

- React – erinevalt Angularist on React tegelikult mitte täisfunktsionaalne raamistik vaid hoopis kasutajaliidese raamistik, mistõttu on sellel küll vähem funktsionaalsusi kuid selle eeliseks on raamistiku kompaktsus [28].

- Vue.js – Javascripti raamistik, mis hakkas populaarsust koguma tänu oma lihtsusele üles seada ühelehelisi veebilehti (ingl. k *Single Page Application*) [28].

Kõik eelnimetatud raamistikud on avatud lähtekoodiga ning nende dokumentatsioon on veebis lihtsasti kättesaadav. Populaarsuse järgi järjestatuna on raamistike loend järgnev: 1. React, 2. Angular, 3. Vue.js [14].

Töö autoril on kogemusi kõigi eelnimetatud raamistikega. Igapäevatöös kasutab töö autor Angulari raamistikku, kuid kõik isiklikud projektid on töö autor loonud React raamistikuga, kuna sellega on ta arendusprotsess kõige tõhusam ning see asjaolu sobib hästi kokku ka käesoleva diplomitöö piiritletud skoobiga. Eelnevatele põhjendustele tuginedes valis autor klientrakenduse arendamiseks React raamistiku.

3.2.3 Andmebaasi tehnoloogiad

Andmebaasisüsteeme saab liigitada mitmeti:

- Relatsioonilised ja mitterelatsioonilised andmebaasid [29].

- Üksikkasutaja andmebaasid ja mitme kasutaja andmebaasid [30].

- Tsentraliseeritud ja hajusad andmebaasid [30].

Eelnimetatud andmebaasisüsteemide liigid täidavad kindlaid rolle tarkvara arendamisel ning vastavalt arendatava süsteemi nõuetele tuleks kindlasti kaaluda erinevaid variante. Näiteks mikrokontrollerite ja mikroarvutite arendamisel, mis vajavad andmebaase, tasub kindlasti vaadata üksikkasutaja andmebaaside poole [31], mitte hajusate andmebaaside poole. Veebiserverite loomisel, mida antud diplomitöö arenduse osa sisaldab, on aga kõige populaarsem liik relatsiooniline andmebaas [32].

„Relatsioonilise andmemudeli esitas 1970 aastal Edgar Frank „Ted“ Codd“ [33]. Ta kirjutas relatsiooniliste andmebaaside kohta artikli, mis tol ajal jäi teiste jaoks märkamatuks. Ta korraldas sümpoosioni, kus veendi IBM rahastama uurimisprojekti

„System R“, millest sai esimene relatsioonilise andmebaasi prototüüp, mille järel arenes välja SQL. [34]

Eelnevale tuginedes on selge, miks relatsioonilised andmebaasid tänapäevalgi laialdaselt kasutusel on: need on loodud aastakümneid tagasi ja need on loomisest saati hästi toiminud. Seetõttu on ka käesoleva töö autor otsustanud relatsioonilise andmebaasi kasuks. Järgnevalt on välja toodud mõningad relatsioonilised andmebaasid.

- Oracle Database – tasuline andmebaasimootor, mida arendab ettevõte Oracle. Jaanuar 2022 seisuga maailmas populaarsuselt esikohal [35]. Kasutatakse laialdaselt suurettevõtetes.
- PostgreSQL – avatud lähtekoodiga relatsiooniline andmebaasimootor [36].
- SQLite – C programmeerimiskeeles kirjutatud, väike, kiire andmebaasimootor. SQLite'i on tõenäoliselt kasutatud rohkem kui kõiki teisi andmebaasimootoreid kokku. [37]

Erinevatel andmebaasimootoritel on küll palju erinevusi, kuid autori arvates täidavad nad veebiserveri arendamisel väga sarnast rolli: andmete talletamine. Sellel põhjusel ei näinud käesoleva töö autor põhjust teha põhjalikumat analüüsi andmebaasimootorite erinevuste kohta, vaid tugines oma varasematele kogemustele ning internetiallikatele [38]. Valituks andmebaasimootoriks osutus PostgreSQL. Töö autor puutub sellega kokku igapäevaselt oma töös, samuti on suurem osa autori varasematest arendusprojektidest loodud just PostgreSQL andmebaasimootorit kasutades. Relatsioonilise andmebaasi kasutamise valikut toetab ka töös käsitletava projekti andmemudel, kus on oluline roll andmeobjektide vahelistel suhetel.

3.2.4 Arenduskeskkondade valik

Veebiplatvormi sujuvaks arenduseks on oluline, et arendaja seab üles mugava ning moodsa arenduskeskkonna. Nii väldib arendaja erinevaid probleeme, näiteks nagu koodi kaotamise [39].

Autor liigitab arenduskeskkonnad kolmeks: versioonihaldus, koodi halduskeskkond, arenduskeskkond. Versioonihaldussüsteemi tuleb kasutada, et hallata koodi muutumist ajas. Igast muudatusest koodis jääb maha jälg ning probleemide korral saab arendaja

„kella tagasi keerata“ ja oma vigu parandada [40]. Koodi halduskeskkondade eesmärk on koodi kuskil hoiustada. Koodi saab hoiustada kolmel viisil: Lokaalses arvutis, võrgukettal või pilveserveris [41]. Arenduskeskkond on vajalik, et arendaja saaks kasutada modernseid tööriistu, mis abistavad arendajat koodi kirjutamisel, testide jooksutamisel, pakuvad koodi automaatse sõnalõpetust ja teisi kasulikke funktsioone [42].

Versioonihaldussüsteemidest on autorile tuttavad vaid kaks: Git ja Apache Subversion. Git on vabavara kogukonna poolt väga laialdaselt toetatud ning on kiiresti saanud maailma populaarseimaks versioonihaldussüsteemiks [43]. Apache Subversion on tsentraliseeritud versioonihaldussüsteem. See erineb Git'ist selle poolest, et kui Git'i puhul iga arendaja teeb projektist täieliku koopia, siis Apache Subversioni puhul toimib kõik tsentraliseeritud koodihoidlas kasutades klient-server süsteemi [43]. Antud diplomitöö raames oli arendajaks vaid diplomitöö autor ning koodi teiste arendajatega jagada ei tulnud, seega tulenes versioonihaldussüsteemi valik ainult populaarsusest. Töö autor valis versioonihaldussüsteemiks Git'i.

Koodi halduskeskkondasid on mitmeid, kuid antud töös on autor välja toonud vaid need keskkonnad, millel on olemas tasuta versioonid ning millega autor on ise otseselt kokku puutunud.

- Bitbucket

Bitbucket on Atlassiani loodud Git koodihoidla halduskeskkond mille disain keskendub professionaalsetele tiimidele [44]. Bitbucketis on üksikul arendajal võimalik hoida oma Git koodi repositooriume tasuta [45].

- GitHub

GitHub on sarnane tööriist Bitbucketile, kuid selle halduskeskkonna fookus on selle loomisest saati olnud avatud lähtekoodiga projektide toetamine: selle halduskeskkonna kasutajaliides on kasutajasõbralik. Igaüks võib GitHubiga liituda ning luua oma koodile koodihoidlaid [46].

Tuginedes eelnevale informatsioonile, valis töö autor koodi halduskeskkonnaks GitHubi, kuna diplomitöö käigus valminud veebiplatvormi kood on vabavara ning töö

autor loodab, et tulevikus teised huvilised leiavad tänu sellele keskkonnale selle projekti kiiremini üles, kui nad peaksid tahtma sellesse panustada.

Arenduskeskkonda valides oli töö autoril valik kahe programmi vahel, sest need olid ainukesed keskkonnad, millega ta oli kokku puutunud: IntelliJ IDEA ja Visual Studio Code. IntelliJ IDEA on ettevõtte Jetbrains poolt loodud ja hallatud integreeritud arenduskeskkond, mis keskendub Java ning Kotlini programmeerimiskeelte toetamisele, kuid milles saab tegelikult ka teiste programmeerimiskeelte projekte hallata [47]. Visual Studio Code on avatud lähtekoodiga, Microsofti poolt loodud ja hallatud koodiredaktor. Tegemist pole integreeritud arenduskeskkonnaga, vaid see meenutab oma olemuselt pigem tekstiredaktorit, kuid tänu oma suurepärasele liideste haldussüsteemile, koodi automaatsele lõpetamisele ning sisseehitatud Git võimekusele on tegemist suurepärase arenduskeskkonnaga [48].

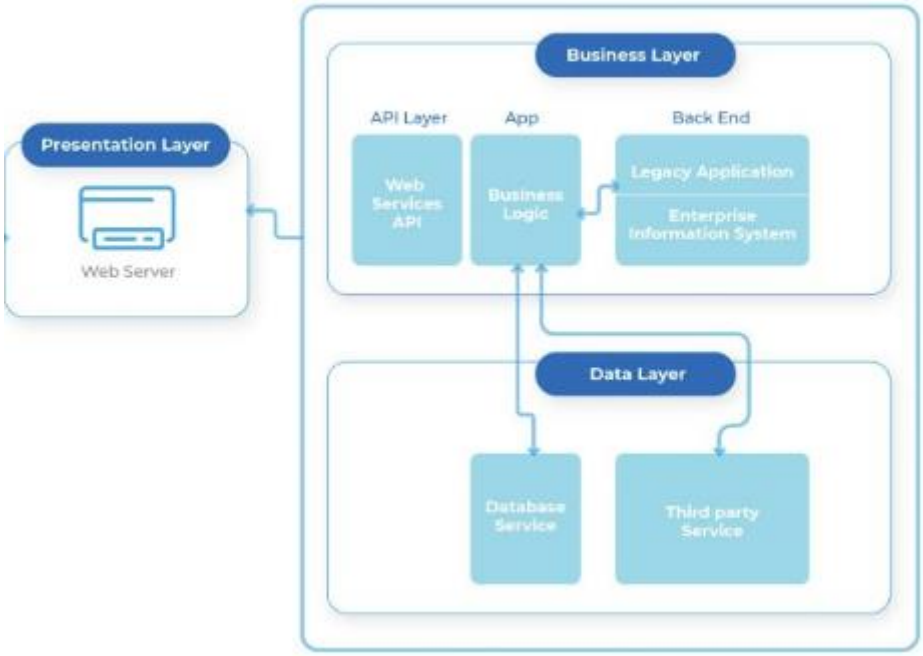
Mõlemad keskkonnad on autori arvates suurepärased tööriistad, kuid tänu IntelliJ IDEA sisseehitatud andmebaasiga ühendamise võimekusele, suurepärasele Spring raamistiku toele ja asjaolule, et IntelliJ IDEA on loodud pidades silmas Java programmeerimiskeelt, mis osutuks veebiserveri arenduse jaoks valituks, valis autor arenduskeskkonnaks just selle tööriista. Siinkohal olgu mainitud, et esimesed kaks esile tõstetud argumenti kehtivad vaid IntelliJ IDEA tasulisele versioonile, mille litsents oli töö autoril olemas kuna ta oli Tallinna Tehnikaülikooli õpilane.

3.3 Veebirakenduse arhitektuur

Veebirakenduse arhitektuuri luues lähtus autor klassikalisest kolmetasandilisest arhitektuurist: esitlustasand – rakendustasand– andmetasand [49]. Kolmetasandiline arhitektuur on laialdaselt levinud ning käesoleva töö autor on sellega koolis ning tööl palju kokku puutunud. Joonisel 1 on välja toodud kolmetasandilise arhitektuuri disain, millele töö autor arendusprotsessi käigus toetus. Joonisel esinevad „*Third party service*“ ning „*Legacy Application*“ osi töö autori koodis ei esine, kuid autor pidas siiski vajalikuks need joonisele jätta, sest tulevikus on tõenäoline, et need osad koodi tekivad.

Kolmetasandilist arhitektuuri ajatakse tihti segi kolmekihilise arhitektuuriga [52]. Kolmetasandilise arhitektuuri puhul on näha, kuidas klientrakenduse osa suhtleb serverirakendusega ning serverirakendus suhtleb andmebaasiga. Selles arhitektuuris on

klientrakendus üks tasand, serverirakendus teine tasand ning andmebaas kolmas tasand. Kolmekihiline arhitektuur, mis on täpsemalt välja toodud ka järgnevas peatükis, keskendub aga hoopis serverirakenduse erinevatele kihtidele.



Joonis 1. Kolmetasandiline arhitektuur [50]

Eksisteerib ka teisi arhitektuurilisi lahendusi, mis pakuvad eelneva lahenduse kitsaskohtadele paremaid alternatiive, kuid antud töö raames ei pidanud autor vajalikuks neid kitsaskohti lähemalt uurida ega lahendada, sest see poleks mahtunud töö skoopi.

3.4 Veebirakenduse disain

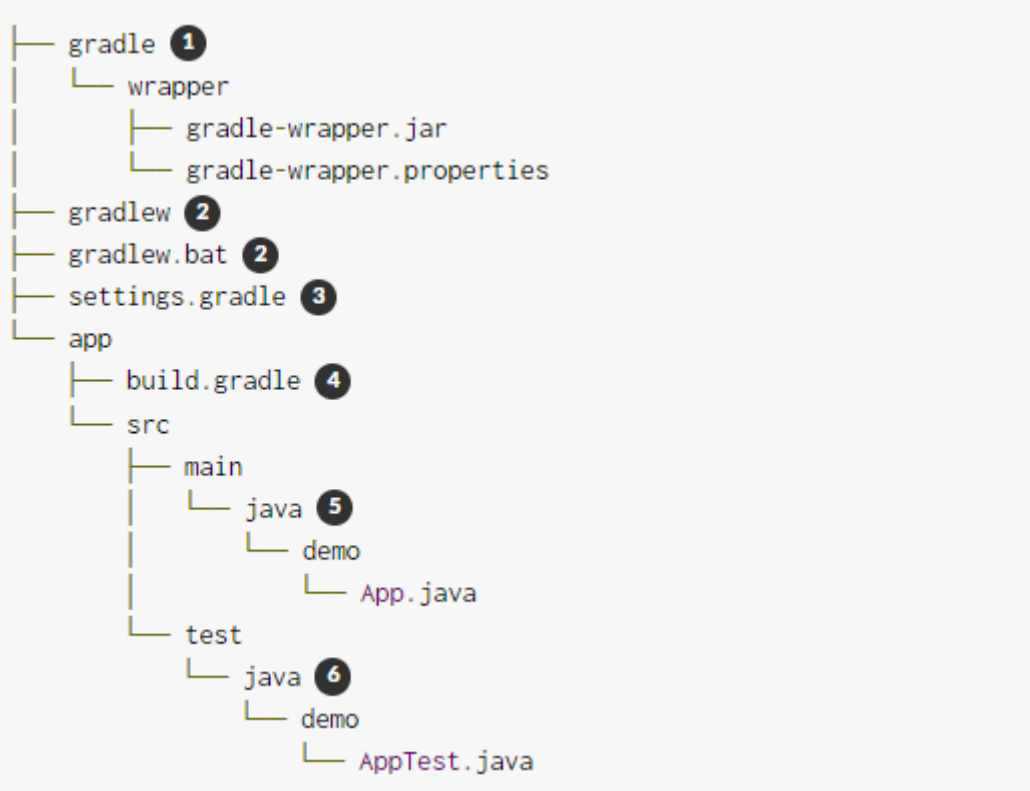
Kogu veebirakenduse ehk nii klientrakenduse kui ka serverirakenduse disainimisel on oluline, et disain oleks selge ja arusaadav, kuna eesmärk on luua avatud lähtekoodiga projekt, mille arendamisse saavad tulevikus ka teised panustada. Järgnevalt käsitletakse serverirakenduse arhitektuurilist disaini ning klientrakenduse puhul nii arhitektuurilist kui ka kasutajaliidese disaini.

3.4.1 Serverirakenduse disain

Esimese sammuna tuleb paika panna kaustade struktuur.

Tavalise java projekti puhul ei esine arendajal ühtegi piirangut, kuidas oma koodi struktureerida. Arendaja võib kõiki oma programmifaile hoida ühes kaustas ning neid eraldiseisvalt jooksutada. Kasutades aga Gradle koodiehituse automatiseerimist, kaob see vabadus arendajal ära. Selleks, et Gradle saaks toimida ning selle tööriista väljund oleks selline, mida arendaja ootab, peab kogu serverirakenduse struktuur vastama Gradle väljapakutud standardile [53].

Gradle välja pakutud standardiseeritud struktuur on välja toodud Joonisel 2.



Joonis 2. Gradle välja pakutud projekti struktuur

Töö autor otsustas järgida Gradle väljapakutud struktuuri. Selle struktuuri üldine põhimõte on, et kogu Java lähtekood asub projektis *src/main/java* kaustas ning kõik java testidega seotud kood asub *src/test/java* kaustas. Joonisel 2 on ka välja toodud asjaolu, et viimased kaks kausta võivad omakorda sisaldada veel kaustasid, et arendaja saaks oma kirjutatud Java koodi loogiliselt kategoriseerida.

Serverirakenduse disainimisel lähtus käesoleva töö autor Spring raamistiku soovituslikust arhitektuurist, mis on kolmekihiline. See koosneb esitluskihist, äriloogika kihist ning andmebaasi kihist [55].

- Esitluskiht

Esitluskihi ülesanneteks REST serverirakendustel on sissetulevad päringud autoriseerida ning need teksti kujult üle viia POJO kujule ning seejärel päring edastada äriloogika kihile [55].

- Äriloogika kiht

Äriloogika kiht, oma nimele kohaselt, tegeleb sisse tulnud päringu töötlemisega. Selles kihis toimub päringute valideerimine, autoriseerimine ning päringutele vastuste kokkupanemine. Äriloogika kihis toimub ka väliste teenuste kasutamine. Äriloogika kihil on tihti vaja saada andmebaasi salvestatud andmeid või andmebaasi uusi andmeid salvestada. Selleks muudab see kiht andmed andmebaasi kihi jaoks sobivale kujule ning edastab andmed. Andmebaasi suhtluskiht vastab äriloogika kihile uute andmetega või informatsiooniga andmebaasipäringute õnnestumistest või ebaõnnestumistest.

- Andmebaasi suhtluskiht

Andmebaasi suhtluskihil on ainult üks vastutus: Andmeid salvestada, uuendada, kustutada ja pärida [55]. Eelnimetatud toimingute kohta öeldakse, et need on CRUD operatsioonid (*Create, Read, Update, Delete*).

Sellise kolmekihilise struktuuri põhiline mõte on *Separation of Concerns*. Seda terminit kasutati esimest korda Edsger W. Dijkstra artiklis „*On the role of scientific thought*“ aastal 1974 [56]. Sellise disainiga jagatakse rakendus „tükkideks“, kus iga tükk täidab mingit kindlat rolli. Eelnevalt on välja toodud iga kihi roll, kasutades terminit „ülesanne“. Oma tagarakendust selliselt disainides ning nendest disainimustritest kinni pidades on suurem tõenäosus, et kui tulevikus on vaja mõni kiht asendada uuega, või lisada uusi kihte, siis see on lihtne. Seda lähenemisviisi on Robert C. Martin autori arvates väga hästi kirjeldanud mitmete heade näidetega oma raamatus „*Clean Architecture: A Craftsman's Guide to Software Structure and Design*“.

Hüpoteetiline näide juhusest, kus üks kiht tuleb asendada uuega, tuginedes eelnevalt välja toodud raamatule on järgnev.

Uuele veebiplatvormile luuakse uus klientrakendus, kuid uus rakendus ei toeta REST API kommunikatsiooni, vaid mingisugust teist standardit, näiteks gRPC. Antud juhul on

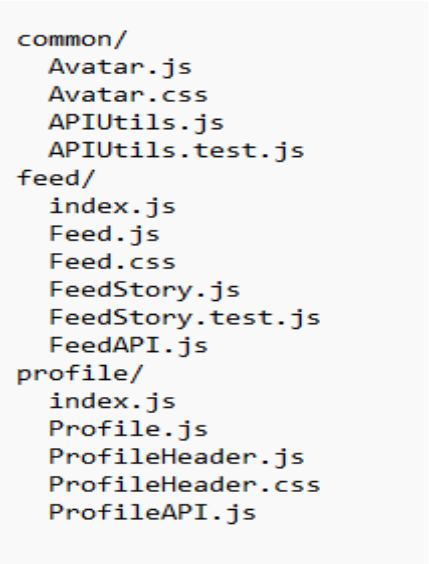
vaja asendada vaid esitluskiht uue vastu, mis toetaks gRPC sõnumite vastuvõtmist ja saatmist, ning suudaks päringud viia äriloogika kihi jaoks vajalikule kujule. Ülejäänud kaks kihti jäävad seejuures puutumata.

3.4.2 Klientrakenduse disain

Serverirakenduse disainimise puhul on üks olulisemaid aspekte projekti struktuur ning erinevate tõestatud disainimustrite kasutamine, et arendusprotsessi käigus efektiivsus ei langeks. Klientrakenduse puhul on olukord veel keerulisem, kuna sellel rakendusel kehtib kõik eelnevalt välja toodud, aga klientrakenduses tähendab „disain“ ka kasutajaliidese disaini. Klientrakenduse puhul pole autori arvates projekti struktuur rangelt sama oluline kui serverirakenduse puhul, kuid on siiski piisavalt oluline, et sellele tähelepanu pöörata. Järgnevalt kirjeldatakse kõigepealt klientrakenduse struktuuri ja disainimustreid ning seejärel kasutajaliidese disaini.

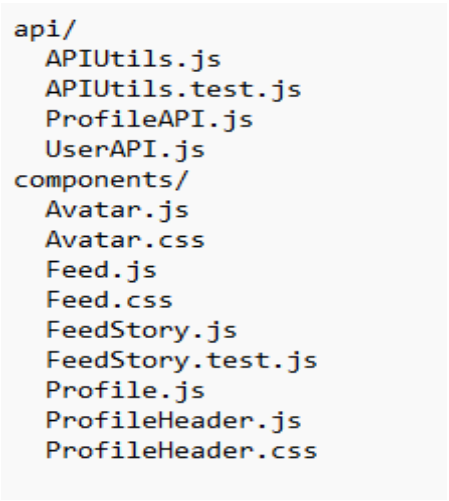
Klientrakendus loodi React raamistikuga. React raamistiku arendajad on dokumentatsioonis välja toonud, et projekti struktuur ei ole React raamistiku jaoks oluline, küll aga on mõned populaarsed lähenemisviisid, mida arendajad on hakanud kasutama [57].

Üheks lähenemisviisiks on failide grupeerimine klientrakenduse loogiliste osade või klientrakenduse erinevate lehtede asukohtade põhjal. Joonisel 3. on välja toodud näide klientrakenduse kaustastruktuurist, kus failid on jagatud kaustadesse klientrakenduste loogiliste osade põhjal. Selgesti on eristatav, et on ühiste osade kaust (*common*), ajajoone kaust (*feed*) ning profiili kaust (*profile*).



Joonis 3. Kaustastruktuur loogiliste osade põhjal

Teiseks lähenemisviisiks on failid jagada kaustadesse failide sarnasuse järgi. Joonisel 4. on välja toodud näide sellisest kaustastruktuurist. Selgesti on eristatav, et kõik API-ga seotud failid on kaustas „*api*“ ning kõik komponentidega seotud failid on kaustas „*components*“.



Joonis 4. Kaustastruktuur failide sarnasuse järgi

Töö autor valis oma klientrakenduse kaustastruktuuri loomisel esimese lähenemise, kuna oli seda lähenemist minevikus kasutanud ka muudes projektides.

Klientrakenduse kasutajaliidese disainimisel tuli silmas pidada asjaolu, et kasutajaliides peab olema kasutatav nii põhikooliõpilastele kui ka nende õpetajatele. Sõltuvalt õpilaste noorest east tegi autor teadliku valiku teha kasutajaliides visuaalselt kaasahaaravaks, intuitiivseks ning eakohaseks kasutades selleks lihtsat navigeerimist, menüüdes, keskmisest suuremat teksti ning kasutades erinevaid värve ning ikoone [58]. Värvide valikul tugines autor Raúl Gómez Acuña blogipostitusele „*Colour your apps in React Native using Material Palette"* [58]. Õpetajate nõuete kohaselt oli esikohal asjaolu, et küsimuste loomine peaks olema intuitiivne ning ei tohiks olla aeganõudev, seetõttu keskendus autor kasutajaliidese disainimisel nende osade efektiivsele disainimisele. Autor üritas paika panna lihtsad menüüelemendid ning kasutaja sisenditele visuaalse hierarhia, et õpetajad saaksid aru, kui kaugel nad oma protsessiga on. Viimase väljatoodud punkti kasulikkuses on autor veendunud Meelis Antoi kursusel „IT süsteemide kasutatavus" ning samale ideele toetub ka Jeff Johnson oma raamatus „*Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*". Sõltumata eelnevatest erinevatest nõuetest jäi autor siiski ka kindlaks oma arvamusele, et kasutajaliides peaks olema ühtlane nii õpetaja kui ka õpetaja vaadetes, tuginedes oma varasematele kogemustele põhikoolis, gümnaasiumis ning ülikoolis, kus õpetajatel oli probleeme õpilaste aitamisega erinevates infosüsteemides, sest nende kasutajaliides oli õpilaste omast totaalselt erinev.

Täpsem kasutajaliidese disain, näited ning mõningad selgitused on välja toodud peatükis „Arendusprotsess".

3.4.3 Andmebaasi disain

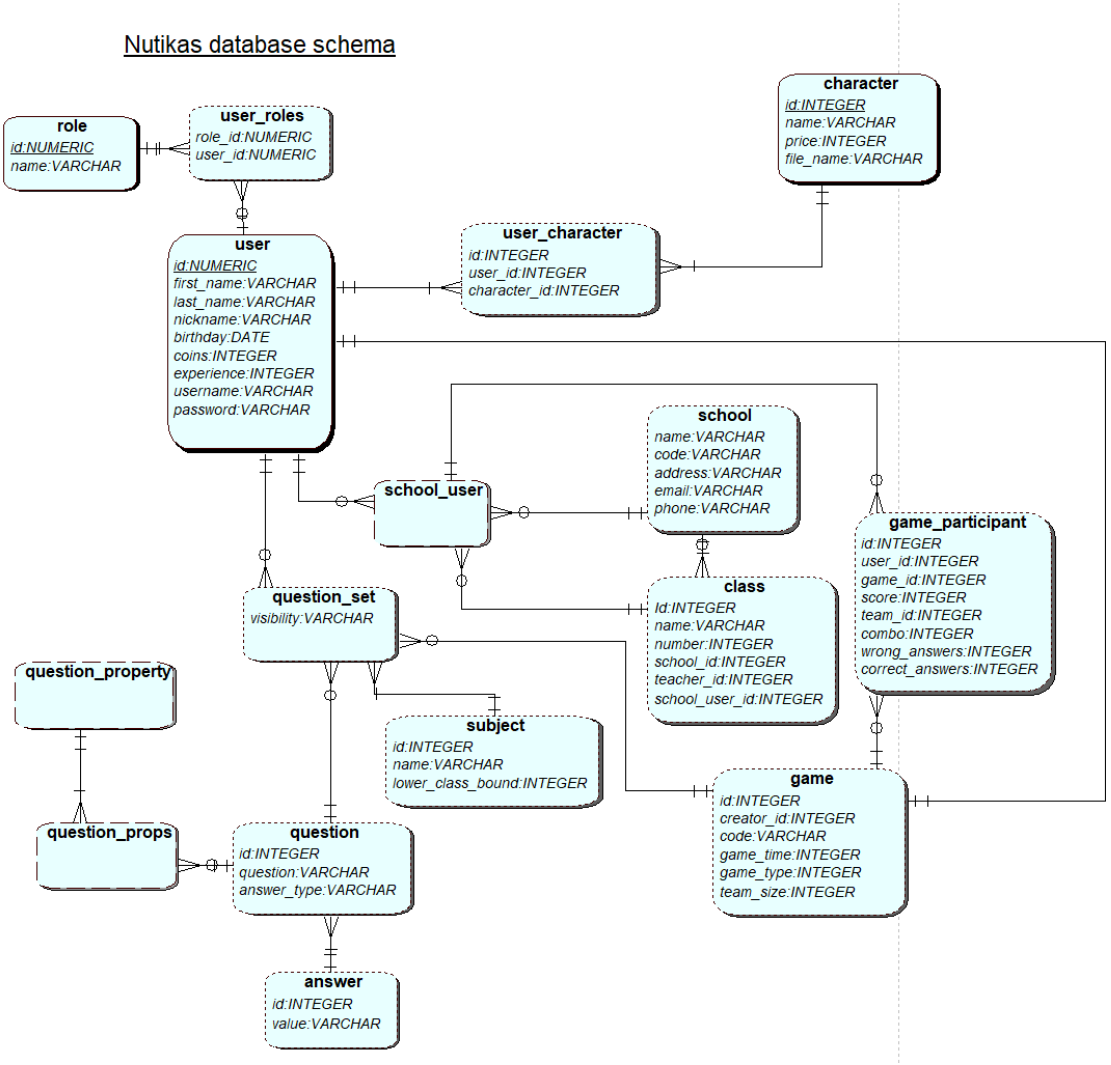
Andmebaasi mudeli koostamiseks on kõige laiemalt levinud metoodika olemi-suhte diagrammi (*ERD – Entity Reliationship Diagram*) loomine, mis on pool-formaalne mudel, millega saab välja tuua andmemudeli graafilise kuju [60]. Olemi-suhte diagramm on suurepärane tööriist just relatsiooniliste andmebaaside kujutamiseks, sest kõik tabelite vahelised seosed on välja toodud joonte abil.

Esialgsel andmebaasi disainimisel piisab sellest, kui on olemid on kontseptuaalsel tasemel, s.t ilma olemi sisemist struktuuri täpsustamata. „Sellist ERD mudelit, kus

olemite sisemine struktuur ei ole veel täpsustunud nimetatakse infoloogilisteks mudeliteks“ [61].

Olemi-suhte diagrammide loomiseks on mitmeid vabavaralisi tööriistasid [62]. Töö autor on kokku puutunud vaid ühega – andmebaasisüsteemide aluste kursusel Priit Raspeli tutvustatud tööriistaga „QSEE SuperLite“. Viimasega on võimalik lihtsa ning intuitiivse kasutajaliidese abil luua olemeid ning nende vahelisi suhteid, et tekiks visuaalne representatsioon loodava andmebaasi kujust, mis lihtsustab tulevikus SQL lausete kirjutamist.

Autor pidas vajalikuks esialgu luua diagramm selliselt, et oleks arusaam andmebaasi tähtsamatest tabelitest, kuid sellega ka piirduti. Andmebaasimuudatusi, mis arenduse käigus vajalikud olid, autor enam olemi-suhte diagrammi ei lisanud. Seda põhjusel, et IntelliJ IDEA arenduskeskkonnas kasutatav andmebaasidega ühendamise tööriist suudab andmebaasist ERD mudeleid automaatselt genereerida. ERD mudel oli autori jaoks vaid abistav vahend esialgsete SQL-lausete kirjutamiseks ning andmebaasitabelite suhete loomiseks. Järgnevalt on joonisel 5 välja toodud arendusprotsessi alguses välja töötatud olemi-suhte diagramm, mis arendusprotsessi jooksul pidevalt muutus.



Joonis 5. Esialgne olemi-suhte diagramm

3.5 Analüüsi kokkuvõte

Analüüsi käigus pandi kirja õpilaste ning õpetajate nõuded loodavale veebiplatvormile, ning nendest loodi kasutajalood. Kasutajalood jaotati funktsionaalseteks ja mittefunktsionaalseteks ning õpilase ning õpetaja kasutajalood toodi eraldi välja. Samuti pandi kirja kasutajalood, mis antud diplomitöö skoopi ei mahtunud, kuid mis on diplomitöö autoril kavas tulevikus luua.

Järgnevalt toodi välja erinevaid tehnoloogiaid jaotades need klientrakenduse tehnoloogiateks, tagarakenduse tehnoloogiateks ning andmebaasi tehnoloogiateks, mille vahel autor pidi tegema põhjendatud valiku. Välja toodi viis tagarakenduse

programmeerimiskeelt koos iga keele jaoks autorile tuttava raamistikuga. Autori valikuks osutus analüüsi tulemusena Java programmeerimiskeel koos Spring raamistikuga. Klientrakenduse tehnoloogiaid toodi välja kolm, kõik olid Javascripti programmeerimiskeele raamistikud, millest valituks osutus React.

Andmebaasimootorite analüüsi käigus vaadati lähemalt kolme relatsioonilise andmebaasi mootorit, Oracle, PostgreSQL ning SQLite. Eelnevatest valis autor kasutatavaks andmebaasimootoriks PostgreSQL ning tõi välja oma põhjendatud arvamuse.

Arenduskeskkonda valides oli autoril valida kahe tööriista vahel, Visual Studio Code ning IntelliJ IDEA. Autoril oli olemas IntelliJ IDEA tasuline litsents, mistõttu osutus see valituks. Koodi versioonihaldussüsteemiks valis autor Git'i, ning oma koodi otsustati hoida GitHub keskkonnas.

Veebiplatvormi rakenduse disainimisel lähtus autor erinevate oma valitud tööriistade dokumentatsioonis välja toodud parimatel praktikatel ning soovitustel. Viimane hõlmas endas kaustastruktuuri kindlal kujul hoidmist, kolmetasandilist suhtlust klientrakenduse, serverirakenduse ning andmebaasi vahel ning kolmekihilist struktuuri serverirakenduses. Andmebaasi disainiprotsess sai alguse olemi-suhtediagrammi loomisega, mille põhjal koostas autor SQL laused.

4 Arendusprotsess

Reeglina on töö autori kogemuste põhjal erinevad projektid alguse saanud just arendusprotsessist. Täpsemalt on populaarne lähenemine, kus peale idee sündimist luuakse kiiresti arendusprotsessiga minimaalselt töötav toode ehk *MVP*. Antud diplomitöö õnnestumiseks tuli siiski teha mahukas analüüs enne arendusprotsessi algatamist, sest töö autor pidas lõputöö mahukuse ning enda ning fookusgrupis osalenud õpetajate ja õpilaste aja planeerimise huvides mõistlikuks mitte läheneda projekti valmimisele agiilselt, vaid koguda eelnevalt nõuded ning seejärel need täita.

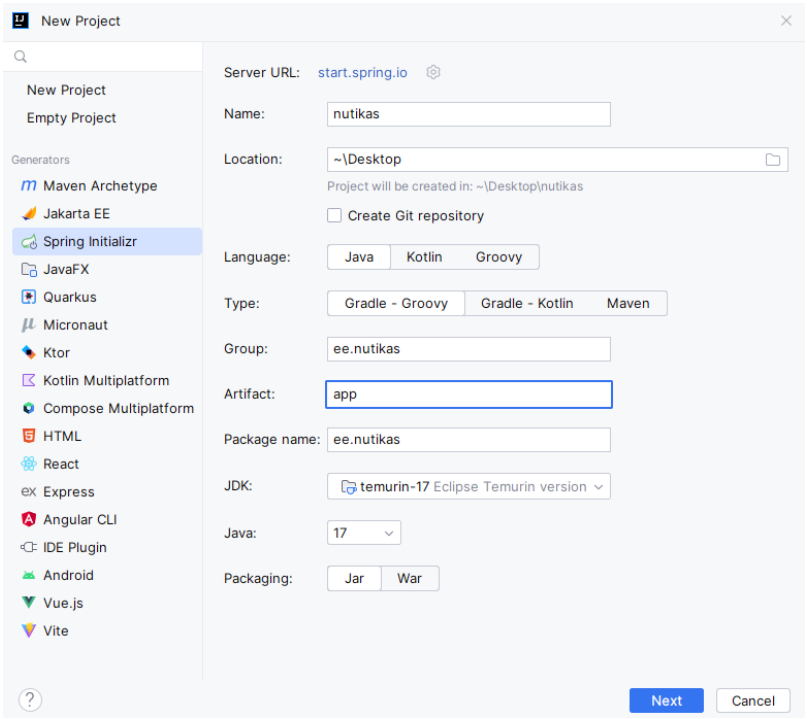
Arendusprotsess on jaotatud kaheks osaks, esimeses osas käsitletakse serverirakenduse arendust ning teises osas klientrakenduse arendust. Mõlema funktsionaalsuse arendus on kajastatud erinevates peatükkides, kuid oluline on mainida, et tegelikult oli kogu arendusprotsessi vältel tegemist paralleelselt mõlema rakenduse arendusega.

4.1 Serverirakenduse arendus

Serverirakenduse arendusprotsess algab projekti, integreeritud arenduskeskkonna ning sõltuvuste seadistamisega. Peale seadistusprotsessi on võimalik hakata tegelema arendusprotsessiga. Järgnevalt on mõlemat lähemalt kirjeldatud.

4.1.1 Serverirakenduse seadistus

Serverirakenduse arendust alustas autor projekti loomisest. Uue Java rakenduse loomiseks, mis kasutab Spring raamistikku, on IntelliJ IDEA'l sisseehitatud tööriist, millele pääseb ligi navigeerides *File -> New -> Project* ning valides vasakpoolsest menüüst „Spring Initializr". Arendajale kuvatakse erinevad sätted, mida saab vastavalt vajadusele ning soovidele muuta. Joonisel 6 on kuvatud konfiguratsioon, mis vastab täpselt sellele, millega autor oma uue projekti lõi.



Joonis 6. Spring Initializr konfiguratsioon

Järgmise sammuna tuli antud tööriistaga valida lisateegid, mida Spring raamistikuga kasutama hakatakse. Autor valis esialgsel projekti loomisel viis lisateeki, mis on järgnevalt eraldi välja toodud.

- Lombok – teek, mille abil saab arendaja vältida trafaretset koodi, kasutades selleks Lomboki poolt defineeritud annotatsioone, mille abil teek java koodi kompileerimise etapis ise genereerib [63].
- Spring Web – Spring raamistiku moodul, mille abil saab arendaja luua REST lähenemisviisiga rakendusliideseid. Spring Web kasutab Apache Tomcat sisseehitatud veebiserverit [64].
- Spring Security – Spring raamistiku moodul, millega saab hallata veebiserveri päringute autoriseerimist ning autentimist [65].
- JDBC API – Andmebaasiga ühendamise rakendusliides, mille abil saab defineerida andmebaasimootori, ühenduse detailid ning SQL käskude loomise viisi [66].
- Flyway Migration – andmebaasi versioonihaldussüsteem, mille kohaselt arendaja lisab oma loodud SQL failid projekti *resources/db/migration* kausta ning projekti jooksutamisel toimub antud SQL failide valideerimine ning andmebaasi migratsioon [67].

Järgmine samm oli projekt luua ning IntelliJ IDEA tegi kõik vajalikud sammud arendaja eest ise ära, ning projekt oli loodud, ning töö autor sai hakata oma projekti arendama.

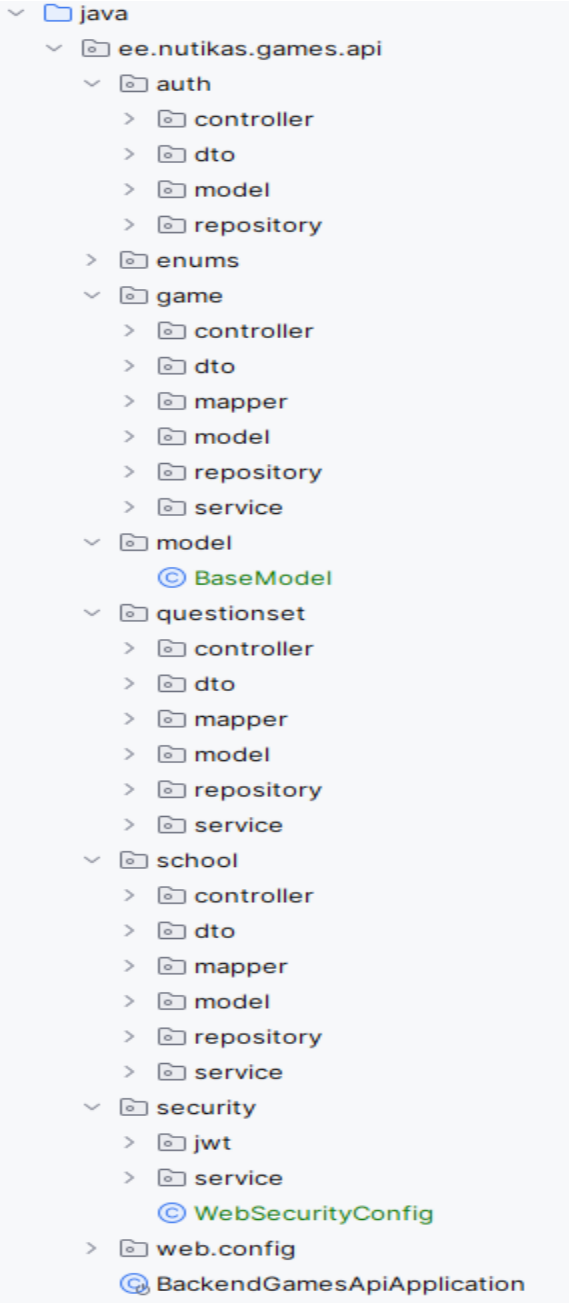
4.1.2 Serverirakenduse struktuur

Eelnevas peatükis käsitleti juba põgusalt serverirakenduse vajalikku struktuuri, et kasutatav tööriist Gradle toimiks ootuspäraselt, ent mainiti, et selles struktuuris sees saab arendaja oma projekti struktuuri ise hallata.

Üks Spring raamistikku kasutatavate arendajate levinud struktuur sarnaneb kolmekihilise arhitektuuri mudelile: loodavad kaustad jagunevad erinevate ülesannete järgi. Näide sellisest struktuurist on järgnev [68]:

- *controller* – kaust, milles hoitakse koodi, milles on REST rakendusliidese kihi ülesannet täitvad java klassid.
- *service* – kaust, milles hoitakse java klasse mis tegelevad vaid äriloogikaga.
- *repository* – kaust, milles oleva koodi ülesanne on andmebaasiga suhelda.
- *config* – kaust, milles esineb kogu projekti java koodil põhinev konfiguratsioon.
- *dto* – kaust, milles esinevad POJO (Plain Old Java Object) klassid, mille ainus ülesanne on andmeid edastada.
- *entity / domain / model* – kaust, milles esinevad java klassid, mis kujutavad endast relatsioonilise andmebaasi domeeniobjekte.
- *Security* – kaust, milles esinevad Spring raamistiku Security mooduli klassid.

Antud struktuur on laialdaselt levinud, kuid autori varasemate arenduskogemuste põhjal on sellisel struktuuril üks suur probleem: suurte arendusprojektide puhul kipub igasse eelnimetatud kausta tekkima piisavalt palju Java klasse, et projekti muutub raskesti hallatavaks. Seetõttu otsustas autor võtta inspiratsiooni klientrakenduse analüüsi käigus ühe alternatiivina pakutud lahendusest: jaotada kaustad funktsionaalsuse põhjal ning iga funktsiooni kaustadesse luua eelnevalt välja toodud struktuur. Erandiks oli *security* kaust, mis jäeti eraldiseisvaks. Tulemuseks oli kaustastruktuur, mis on välja toodud joonisel 7.



Joonis 7. Tagarakenduse kaustastruktuur

Väljatoodud struktuuri eeliseks on asjaolu, et tulevikus funktsionaalsusi lisades luuakse uus kaust, ning olemasoleva funktsionaalsuse klasside kättesaadavus ei ole sellest mõjutatud. Joonisel on kuvatud veebiplatvormi funktsionaalsused kausta põhiselt: autentimine, mängude funktsionaalsus, küsimustike funktsionaalsus, koolide ja kasutajate funktsionaalsus, turvalisus ning konfiguratsioon.

4.1.3 Serverirakenduse turvalisus

REST API päringute autentimiseks otsustas autor kasutada Spring raamistiku Security moodulit. See moodul pakub mitmeid valmiskirjutatud ning vabavara arendajate poolt valideeritud funktsionaalsusi mis on seotud serverirakenduse turvalisusega. Kasutajate autentimiseks on tänapäeval enimlevinud kaks meetodit: sessiooniautentimine ning *token*-põhine autentimine [69]. Nende põhiline erinevus on, et sessiooniautentimise korral hoitakse autentimise detaile serverirakenduse sessioonis aga *token*-põhisel autentimisel hoopis klientrakenduse poolel. Töö autor oli kokku puutunud *token*-autentimisega, sellest lähtuvalt langetas autor otsuse kasutada seda viisi. See autentimisviis kujutab endast klient- ka serverirakenduse suhtlust, kus klient paneb igale päringule kaasa sisse logides saadud *token*'i, mille serverirakendus genereeris ning mida serverirakendus suudab valideerida. Antud raamistik toetab ka klientide autoriseerimist. Autentimine ning autoriseerimine on kaks erinevat põhimõtet: autentimine vastutab kasutajate identiteedi kontrollimise eest, et otsustada, kas kasutajal on juurdepääs serverirakendusele, autoriseerimine on aga protsess, mille käigus otsustatakse, millistele andmetele on kasutajal juurdepääs. Käesolevas töös käsitletakse lähemalt autentimist.

Autentimine seati üles tuginedes tuntud arendusmeetodile nimega *Aspect-Oriented Programming* (AOP) [70]. Selle meetodi idee seisneb rakenduse põhiloogika ning korduvate funktsionaalsete osade eraldamises. Autentimine on korduv osa, mis kehtib üle terve serverirakenduse, seega on põhjendatud otsus hoida seda funktsionaalsust eraldi. Autentimise funktsionaalsus loodi kasutades Spring raamistiku REST API päringute filtreerimise funktsionaalsust. Iga sissetulev päring suunati kõigepealt autentimise filtrisse, mille ülesanne oli päringuga kaasa saadedud *token*'i valideerimine ning seejärel päring edasi suunata või teavitada päringu saatjat autentimise veast.

Token'itega seotud loogika jaoks kasutas autor jsonwebtoken raamistikku, millesse on juba kõik vajaminevad funktsionaalsused sisse ehitatud. Antud raamistik genereerib *token*'eid vastavalt kasutaja sisse antud parameetritele, seega saab selle raamistiku abil genereeritud *token*'itega edastada kogu vajaminevat infot ning tulevikus vajadusel ka infot juurde lisada. Raamistiku nimi tuleneb asjaolust, et enne *token*'i loomist viiakse info JSON formaati ning alles siis genereeritakse sellest token. Nii saab klientrakendus *token*'ist info JSON formaadis kätte, ning kuna JSON formaat on levinud viis andmeedastuseks, siis on seda raamistikku mugav kasutada.

Jsonwebtoken raamistik vajas toimimiseks Spring konfiguratsioonis muudatusi. *Token*'ite genereerimiseks vajas raamistik sõne, mille abil genereeritakse igale *token*'ile *signature* ehk allkiri. Autor genereeris sõne ise ning salvestas selle Spring konfiguratsioonifaili *application.properties.* Samuti vajas raamistik aega, millal *token*'id aeguvad. Autor sätestas selleks ajaks 10 minutit. Joonisel 8 on välja toodud Spring konfiguratsiooni võti-väärtus paarid, mida jsonwebtoken raamistik oma toimimiseks vajas.

```
jwt.secret=/*SECRET HIDDEN FOR SECURITY PURPOSES*/
jwt.expiration=1000000
```

Joonis 8. Jsonwebtoken raamistiku konfiguratsioon

Serverirakendus eeldab, et kui klientrakendus esitab päringuid, mis nõuavad autentimist, siis serveri poolt genereeritud *token* peab sisalduma HTTPS päringu päises võtmega „*Authorization*" ning selle väärtus peab algama sõnega „*Bearer*" millele järgneb tühik ning seejärel *token*'i väärtus.

4.1.4 Andmebaasi ning andmebaasiühenduse loomine

PostgreSQL andmebaasi loomiseks kasutati tööriista Docker, kuna selle tööriista abiga ei pea arendaja oma arvutisse alla laadima PostgreSQL andmebaasimootorit, vaid saab lihtsa *docker-compose.yml* failiga luua andmebaasi loomise konfiguratsiooni ning seda konfiguratsiooni Dockeris jooksutada, mille tulemusena loodakse uus PostgreSQL andmebaas vastavalt kasutaja loodud konfiguratsioonile [71] [72]. Joonisel 9 on välja toodud autori loodud konfiguratsioonifail, mille abil loodi PostgreSQL andmebaas.

```yaml
version: '3'

services:
  postgres:
    image: postgres:14.2
    container_name: backend-games-api-postgres
    ports:
      - "5434:5432"
    environment:
      - POSTGRES_DB=nutikas
      - POSTGRES_USER=nutikas
      - POSTGRES_PASSWORD=/* PASSWORD REMOVED */
```

Joonis 9. Andmebaasi loomise konfiguratsioonifail

Loodud andmebaasi saab jooksutada kasutades Docker Desktop programmi, millel on lihtne kasutajaliides, või kasutades Dockeri käsurea käsku *docker run*. Töö autor kasutas andmebaasi jooksutamiseks vaid oma isiklikku arvutit.

Serverirakenduse ühendamiseks eelnevalt loodud andmebaasiga oli taaskord vaja muuta Spring raamistiku konfiguratsioonifaili *application.properties*. Täpsemalt oli Spring JDBC raamistiku dokumentatsiooni kohaselt vaja lisada konfiguratsiooni kolm võtit ning määrata neile väärtused: *spring.datasource.url*, *spring.datasource.username*, *spring.datasource.password* [66]. Kõik võtmed on seotud andmebaasi ühenduse loomise parameetritega. Viimased kaks võtit said väärtuse andmebaasi loomise käigus valitud väärtustest, esimese võtme väärtuseks määras autor *jdbc:postgresql://localhost:5434/nutikas*, mis sisaldab endas andmebaasiga ühendamise meetodit, andmebaasi domeeni ning porti ning viimases osas ühendatava andmebaasi nime. Peale nende väärtuste lisamise oli vaja antud raamistikul veel PostgreSQL draiveri teegi olemasolu, mis tuli projekti Gradle sõltuvustes eraldi defineerida.

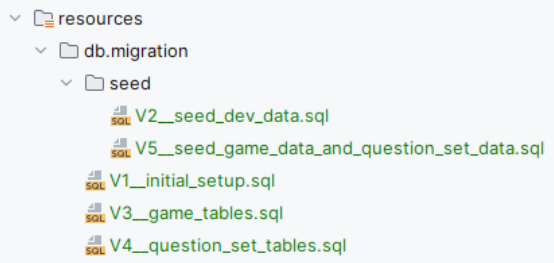
Eelneva etapiga oli serverirakenduse ühenduse loomine andmebaasiga lõpetatud, kuid autor soovis valideerida ühenduse olemasolu seadistades andmebaasi versioonihaldussüsteemi Flyway raamistiku abil. Selleks lisas autor projekti *resources/db/migration* kausta SQL faili nimega *V1_initial_setup*, mis sisaldas vaid ühte SQL käsku: „*SELECT 1*“. Projekti jooksutades eeldas autor, et ühtegi viga ei teki ning programm läheb käima, mis oli ka saavutatud tulemus.

Flyway raamistiku kasutamiseks tuleb arendajal oma loodud SQL skriptid salvestada eraldi SQL failidesse, mille nimetus peab sisaldama suurt V tähte, versiooni numbrit, kahte alakriipsu ning seejärel vabatahtlikku SQL faili kirjeldust [67]. Peale rakenduse käivitamist jooksutab Flyway raamistik kõik SQL skriptid, mis vastavad eelnevalt kirjeldatud nõuetele. See raamistik jooksutab iga SQL skripti vaid ühe korra, järgnevatel programmi käivitamistel vaid valideeritakse jooksutatud skriptide olemasolu andmebaasis. Selleks loob raamistik andmebaasi uue tabeli nimega „*flyway_schema_history*“. Kui arendaja tahab, et raamistik mingisuguse SQL skripti uuesti jooksutaks, peaks ta sellest tabelist vastavad kirjed ära kustutama.

4.1.5 Andmemudeli loomine

Analüüsi viimases osas välja toodud olemi-suhte diagrammile toetudes tuli järgmiseks luua andmemudel. Selleks oli autoril kaks võimalust: koostada SQL käsud manuaalselt ning need Flyway raamistikuga andmebaasis jooksutada, või luua andmemudel kasutades Java klasse ning kasutades ära Hibernate raamistiku funktsionaalsust luua andmebaasi tabelid vastavalt java klassidele ning nendele lisatud annotatsioonidele [73]. Hibernate raamistik on ORM tööriist, mis võimaldab arendajal annotatsioonide ning konfiguratsiooni abil luua Java klasse, mis vastavad serverirakenduse ühendatud relatsioonilise andmebaasi tabelitele. Hibernate raamistik järgib JPA nõudeid. JPA on spetsifikatsioon, mis kirjeldab, millised java klassid kujutavad endast andmemudelit [74]. Mõlemad eelnimetatud tööriistad vajavad toimimiseks JDBC raamistikku, mille autor seadistas projekti loomise käigus. Hibernate täidab Eelnevalt väljatoodud võimalus genereerida andmebaasi tabelid vastavalt arendaja loodud Java klassidele on võimalik kasutades lisades Spring konfiguratsioonifailis võti-väärtus paari „*spring.jpa.hibernate.ddl-auto=true*“. Sellisel andmebaasi genereerimisel on autori arvates probleemikoht: kui arendaja on teinud annotatsioonide lisamisel vea, unustanud mõne annotatsiooni või teinud kirjavea, siis see raamistik seda ei tuvasta, vaid loobki vigaste andmetega tabelid. Selle tõttu otsustas töö autor luua andmebaasi tabelid ning nende vahelised seosed manuaalsete SQL käskudega, ning kasutada hoopis Hibernate raamistiku funktsionaalsust, mis valideerib, et andmebaasi struktuur vastab Java klassides defineeritud struktuurile. Sellist valideerimist võimaldab konfiguratsioon „*spring.jpa.hibernate.ddl-auto=validate*“.

Manuaalselt loodud SQL käsud lisati Flyway raamistiku poolt ettenähtud kausta ning programmi käivitamisel jooksutati need SQL käsud. Joonisel 10 on näha projekti käigus manuaalselt genereeritud SQL skripte. Joonisel on näha, et autor lõi ka kaks SQL skripti, mille abil sisestati andmebaasi tagarakenduse testimiseks mõeldud andmeid, mis asuvad alamkaustas *seed*. Esimeses SQL skriptis loodi kasutajate ja rollidega seotud tabelid ning seosed, kolmandas SQL skriptis loodi mängudega seotud tabelid ning neljandas skriptis küsimustikega seotud tabelid.



Joonis 10. Projekti käigus jooksutatud SQL skriptid

Järgnevalt on joonisel 11 toodud näide Java klassist, mis kasutab Hibernate raamistiku annotatsioone, et luua Javas andmebaasi tabelit kujutav objekt. Klassis kasutatakse annotatsioone nagu

```
@Entity
@Table(name = „school“)
@Column(name „ address“)
```

Joonis 11. Hibernate annotatsioonid

Nende Hibernate annotatsioonide põhjal suudab raamistik teostada andmemudeli valideerimist. Kõik loodud Java klassid, mis kujutavad loodud andmemudelit, on alamklassid klassist BaseModel.java, milles on defineeritud iga andmebaasi tabeli kolm tulpa: automaatselt andmebaasi poolt genereeritud arvuline primaarvõti, mille nimi on „*id*“, automaatselt andmebaasi poolt genereeritud kuupäev ja kellaaeg kirjete sisestamise ning uuendamise kohta nimedega „*created_at*“ ja „*updated_at*“. Järgnevalt on joonisel 12 näiteks toodud koolide informatsiooni talletamiseks vajaliku andmebaasi tabeli Java koodi klass, milles on kasutatud Hibernate annotatsioone.
Käesolevas töös loodud andmebaasi olemid on eraldi välja toodud Lisas 2.

```java
@Entity
@Table(name = "school")
@Getter
@Setter
@ToString
@RequiredArgsConstructor
public class SchoolModel extends BaseModel {

    @Column(name = "name")
    private String name;

    @Column(name = "short_name")
    private String shortName;

    @Column(name = "address")
    private String address;

    @Column(name = "email")
    private String email;

    @Column(name = "phone")
    private String phone;

}
```

Joonis 12. Andmebaasi tabelit "school" kujutav Java klass

4.1.6 Andmebaasi suhtluskihi loomine

Eelnevas peatükis loodud Java klasside objektidele, mida nimetatakse tavapäraselt *Entity*'teks või *Model*'iteks, tuleb andmebaasi suhtluskihis anda väärtused. Iga selline loodud objekt vastab ühele andmebaasis olevale andmereale. Nendele objektidele andmebaasist väärtuste määramiseks on vajalik luua andmebaasi suhtluskihti java klassid, mida autor kutsub *Repository*'deks ehk repositooriumiteks.

Projekti loomise käigus valitud JPA raamistikuga kaasneb juba palju sisseehitatud funktsionaalsuseid, mida arendaja saab kasutada, kui ta laiendab klasse, mis seda funktsionaalsust sisaldavad. Näiteks tuleb eelnimetatud raamistikuga kaasa järgnev liides:

```
JpaRepository<T, ID>
```

Seda liidest laiendades on arendajal võimalik kasutada mõningaid meetodeid, mille abil saab andmebaasist pärida andmeid või neid sinna salvestada, ilma ühtegi rida koodi kirjutamata. Antud liides kasutab parametriseeritud tüüpe, mida näeb eelnevas näites linnunokkade vahel. Parameeter T peab olema JPA *@Entity* annotatsiooniga märgitud Java klass, ning parameeter ID peab olema sama tüüpi objekt, mis on defineeritud

esimese objekti primaarvõtmena. Järgnevalt on joonisel 13 välja toodud näide ühest andmebaasi suhtluskihi liidesest, mis kasutab JPA raamistiku JpaRepository liidest.

```
@Repository
public interface QuestionSetRepository extends
JpaRepository<QuestionSetModel, Long> {

    List<QuestionSetModel> findAllByOwnerId(Long ownerId);

}
```

Joonis 13. Küsimustikkude andmebaasi suhtluskihi liides

Selles koodiosas ilmneb kaks olulist aspekti, miks JPA raamistik on hea tööriist. Esiteks, autor saab defineerida oma loodud liidesesse uusi meetodeid, ning kui need meetodid vastavad JPA meetodite nimetamise spetsifikatsioonile, siis ei ole vaja arendajal kirjutada ühtegi rida SQL-i, vaid see genereeritakse vastavalt meetodi nimele automaatselt. See on lihtsate päringute puhul hea tööriist, et vältida SQL-i kirjutamist. Teiseks, ja autori arvates veel olulisemaks aspektiks, repositooriumid võivadki jääda liidesteks, s.t arendaja ei pea looma ühegi liidese konkreetset implementatsiooni, kui ta seda teha ei soovi. Viimane asjaolu täidab väga hästi Robert C. Martini 2000. aasta artiklis „Design Principles and Design Patterns“ väljatoodud SOLID põhimõtetes oleva „Dependency Inversion“ põhimõttega, mille tähendus seisneb selles, et kood peaks sõltuma abstraktsioonidest, mitte konkreetsetest implementatsioonidest [75]. Kui arendaja ühtegi liidest manuaalselt ei ole implementeerinud, teeb Spring raamistik programmi käivitamisel selle automaatselt arendaja eest ära.
Arendusprotsessi käigus lõi autor igale andmebaasi olemi tabelile repositooriumi, mida ta vastava andmebaasi tabeli olemitega toimetamiseks kasutab.

4.1.7 Äriloogika kihi loomine

Äriloogika- ehk teenusekihi loomisel tegi autor igale veebiplatvormi funktsionaalsele osale eraldi Java klassi. Kõikidele äriloogika klassidele märgiti peale Spring raamistiku annotatsioon *@Service*, mis viitab, et tegemist on teenuseklassiga. Nendesse teenuseklassidesse lõi autor meetodid, mis tegelevad kogu veebiplatvormi funktsionaalsuste realiseerimisega. Iga funktsionaalsuse teenuseklassis on kindlasti üks klassimuutuja: selle sama funktsionaalsusega seotud olemi repositoorium. Nende muutujate väärtused genereeritakse programmi käivitamisel, kui Spring raamistiku *Dependency Injection* ehk sõltuvuse sisestamise funktsionaalsus vastavad repositooriumi implementatsioonid genereerib.

Teenusklasside loomisel on mitu lähenemist, mida arendajad kasutavad. Ühe populaarse lähenemise kohaselt võib ühes teenusklassis olla kasutusel mitmeid repositooriumeid, teise populaarse lähenemise kohaselt peaks igal teenusklassil olema kasutusel vaid üks repositoorium, ning teiste repositooriumi kasutamiseks tuleks teenusel suhelda teiste teenusklassidega. Töö autor eelistab viimast lähenemist, kuna see lähenemine lihtsustab tulevikus mikroteenustele üleminekut, kui see peaks vajalik olema. Kuna autor jagas teenusklassid funktsionaalsuse järgi, siis on loogiline järeldada, et iga teenusklass võib olla tulevikus eraldi mikroteenus, ning seetõttu ei ole mõistlik, kui üks teenus toimetab teise teenuse andmemudeliga.

Teenusklasside loomisel on oluline veel ära märkida DTO ja *Entity*'te vaheline konverteerimine. Andmebaasi suhtluskihi jaoks on oluline vaid Entity'tega toimetamine, sest see kiht annab andmeid välja autori loodud teenusele, kuid teenusklassidest lähevad andmed välja teistesse teenusklassidesse ning API kihti, ning selle jaoks on vajalik, et äriloogika suudab teha *Entity* objektidest DTO objektid. Viimane on tarkvaraarenduse kogukonnas levinud praktika, mida Martin Fowler mainis oma raamatus „Patterns of Enterprise Application Architecture" juba aastal 2002. Selle lähenemisviisiga saavad tarkvaraarendajad lihtsasti hallata, milliseid andmemudeli väljasid nad tahavad avalikustada ning millised andmeväljad on mõeldud vaid teenusesiseseks kasutamiseks.

Selle asemel, et kirjutada konverteerimismeetodeid manuaalselt, on autor kasutusele võtnud Mapstruct raamistiku, mille abil saab kahe Java klassi objekte automaatselt konverteerida. Mapstructi kasutamiseks tuleb luua iga DTO objekti jaoks uus liides ning annoteerida see @Mapper annotatsiooniga. Järgnevalt on joonisel 14 toodud näiteks üks projekti käigus valminud konverteerimisliides.

```
@Mapper
public interface AnswerMapper {
    @Mapping(source = "text", target = "value")
    @Mapping(target = "question", ignore = true)
    AnswerModel toModel(AnswerDTO dto);

    AnswerDTO toDTO(AnswerModel model);
    List<AnswerDTO> toDTOs(List<AnswerModel> models);
}
```

Joonis 14. Mapstruct raamistiku konverteerimisliides

Antud raamistik pakub mitmeid võimalusi defineerimaks, milliseid välju ignoreerida, millised on sissetulevate ning väljaminevate objektide väljade nimetused ja teisi funktsionaalsusi, mida antud diplomitöös lähemalt ei käsitleta.
Teenuskihi arenduse käigus kasutas töö autor iga kord Mapstruct raamistikku, kui ühest teenuskihist oli vajalik teise teenuskihi andmeid või kui edastati andmeid API kihile.
Äriloogikat tehti ainult DTO objektidega, ning alles andmebaasi salvestamisel konverteeriti DTO objektid *Entity*'teks.
Äriloogika kihi testimiseks loodi igale *public* ehk avalikule meetodile ühiktestid. Ühiktestid loodi *src/test/java* kausta täpselt sama struktuuriga nagu äriloogika koodi struktuur, nii on tulevikus lihtne leida üles iga meetodi kohta seda testiv kood.

4.1.8 API kihi loomine

Äriloogika kihi loomisel tuli sarnaselt eelmistele etappidele igale funktsionaalsusele luua uus klass. API kihi klasside nimetus Spring raamistiku soovitusel peaks sisaldama sõna *Controller*. Samuti tuleks kõik API kihis olevad klassid annoteerida Spring raamistiku annotatsiooniga @RestController, kui luuakse REST lähenemisviisiga API-sid. *@RestController* on annotatsioon mis on loodud arendajate mugavuseks, ning asendab *@Controller* annotatsiooniga API kihi klasse, ning see annotatsioon loob funktsionaalsuse, kus Spring raamistik paneb kõigile sellesse API klassi kuuluvate meetodite tagastatavad objektid päringu vastuse *Body*'sse ehk kehasse.

Teine oluline annotatsioon on *@RequestMapping*, mis Spring raamistiku API kihi klassidel peab olema, et selles klassis olevad meetodid oleksid Spring raamistikku sisseehitatud veebiserveri poolt avalikustatavad. Viimane annotatsiooni võtab sulgudesse sõne. Spring raamistikku sisseehitatud veebiserver kogub kõik selle annotatsiooniga sõned kokku ning loob omale ressursitabeli. Selle tabeli põhjal suudab ta sissetulevad päringud suunata õigetesse API klassidesse [76]. Levinud tavaks on nendes sõnedes kasutada ka API versioneerimist. See tähendab sisuliselt seda, et sõned algavad eesliitega *„/api/v"*, kus tähele „v" järgneb API versioon. See on tavaks saanud seetõttu, et aja jooksul äriloogika kiht areneb ning on võimalik olukord, kus äriloogika muutmise tõttu on arendajad sunnitud API kihti muutma. API kihi muutmisel ei ole garanteeritud, et juba olemasolevad API-t kasutavad klientrakendused jäävad töökorda [77].

Igale API kihis olevale meetodile tuleks peale märkida Spring raamistiku annotatsioon, mis vastab API kihi HTTP päringu tüübile. REST API päringu tüübid on eelnevalt välja toodud, nendeks on GET, DELETE, PUT, POST, PATCH jt. Nendele vastavad Spring raamistiku annotatsioonid on @GetMapping, @DeleteMapping, @PutMapping, @PostMapping ja @PatchMapping. Antud annotatsioonid võtavad samuti sisse valikulise sõne, mis täpsustavad omakorda veebiserverile, millise ressursi juurde mingi päringu korral tuleks suunata.

API kihi kõigil meetoditel on vaid üks eesmärk: päringu edastamine teenuskihti, ning vajadusel teenuskihilt saadud päringu vastuse tagastamine päringu saatjale.

Lõpetuseks on oluline märkida, et siin peatükis on välja toodud vaid käesoleva diplomitöö autori arvamusel põhinev kõige olulisem loogika. Spring raamistiku funktsionaalsus on liiga kompleksne, et seda täies mahus käsitleda bakalaureusetöös ning autor soovitab lugejal parema ülevaate saamiseks lugeda Spring raamistiku ametlikku dokumentatsiooni.

4.2 Klientrakenduse arendus

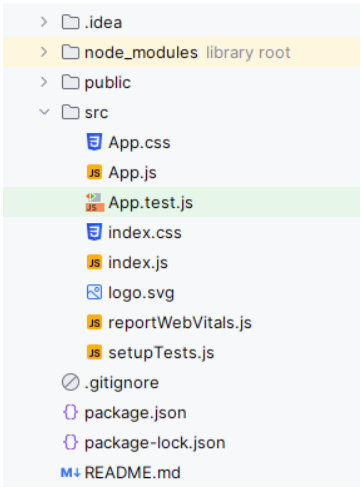
Sarnaselt serverirakendusele on ka klientrakenduse arenduse esimeseks etapiks hoopis seadistus. Erinevalt serverirakenduse arendusest on aga klientrakendusel arendusprotsessi käigus ka luua kasutajaliides, mille arendamine nõuab hoopis teistsuguseid teadmisi kui serverirakenduse arendus. Siiski on oluline, et ka klientrakenduse arendamise käigus peetakse kinni levinud aktsepteeritud tavadest koodi struktuuri osas. Järgnevalt on kõik eelnimetatud protsesside käik välja toodud.

4.2.1 Klientrakenduse seadistus

Klientrakendus on loodud Javascripti raamistiku React abil. Uute React projektide seadistuse efektiivsemaks tegemiseks on Facebooki arendajad loonud tööriista Create React App, mille abil on võimalik käsureal luua uus projekt, kus seadistused tehakse automaatselt. Create React App tööriista kasutamiseks on vajalik, et arendajal on arvutisse alla laetud Javascripti käituskeskkond Node 14.0.0. Selle tööriistaga tuleb kaasa teegihaldusprogramm Node Package Manager, mille abil on võimalik luua uus React raamistiku javascripti projekt.

```
npm init react-app nutikas --template typescript
```

Antud käsk loob uue javascripti projekti, milles on React raamistiku teegid automaatselt seadistatud ning arendajal on võimalik kohe arendusprotsessiga peale hakata. Järgneval joonisel on välja toodud Create React App abil loodud projekti esialgne ülesehitus.



Joonis 15. Create React App tööriistaga loodud projekti ülesehitus

4.2.2 Klientrakenduse ühendamine serverirakendusega

Eelnevalt on välja toodud, et töö käigus valminud klientrakendus ning serverirakendus suhtlevad omavahel HTTP päringutega REST lähenemisviisil. Selleks kasutas töö autor oma klientrakenduses javascripti sisseehitatud *Fetch* API-t, mille abil saab vähese konfiguratsiooniga teha HTTP päringuid. Antud funktsionaalsuse kasutamine on vajalik, et klientrakenduses saaks kuvada serverirakenduses hallatud ning talletatud andmeid.

Klientrakenduses tehtavad päringud saab jaotada kaheks: autentimata ning autenditud päringud. Autentimata päringud on tavalised HTTP päringud kasutades *fetch* API-t, autenditud päringud sisaldavad endas serverirakenduse poolt nõutud HTTP päringu päises asuvat *Authorization* võtmega *token*'i väärtust. Viimase saab klientrakendus määrata aga alles peale seda, kui on tehtud õnnestunud sisselogimise päring, mille vastuse kehas olev *token* tuleb järgnevate päringutega kaasa panna. Järgnevalt on välja toodud kaks koodinäidet, joonisel 16 on autentimata päringu tegemine sisselogimiseks ning joonisel 17 on autenditud päringu tegemine.

```
const data = await fetch('http://localhost:8080/api/auth/login', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
       nickname: username,
       password: password,
    }),
});
```

Joonis 16. Autentimata päringu kood sisselogimiseks

Eelnevas koodinäites on näha, et luuakse POST päring, millel on päises *Content-Type* ehk sisu tüübi võti, mille väärtus on „*application/json*“, mis on standard REST lähenemisviisiga HTTP päringute tegemisel. Päringu kehasse on pandud JSON kujul kasutajanimi ning parool, millega sisse üritatakse logida.

```
public async fetchWithAuth(input: RequestInfo, init?: RequestInit):
Promise<Response> {
    const requestInit = init ? { ...init } : {};
    if (this.tokens) {
       requestInit.headers = {
          ...requestInit.headers,
          Authorization: `Bearer ${this.tokens.accessToken}`,
          'Content-Type': 'application/json',
       };
    }

    const response = await fetch(input, requestInit);

    return response;
}
```

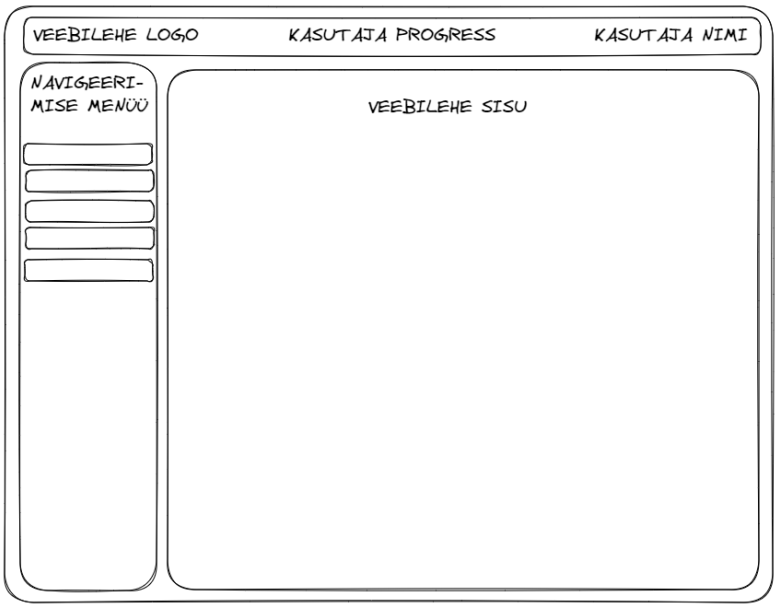
Joonis 17. Autenditud päringu tegemine

Siin koodinäites on välja toodud autori poolt klientrakendusse loodud abimeetod, mille abil saab teha esimeses näites väljatoodud päringuid, millele lisatakse juurde sisselogimisel saadud *token*.
Lisas 3 on välja toodud kõik klientrakenduses tehtavad HTTP päringud.

4.2.3 Klientrakenduse kasutajaliidese loomine

Kasutajaliidese loomiseks tuli autoril kõigepealt teha disainiotsus loodava veebisaidi paigutuse kohta. Veebilehe kasutusmugavus, õpitavus, meelde jäävus, disain ning efektiivsus sõltuvad kõik sellest, millise paigutuse arendaja valib. Erinevalt varasemalt välja toodud kokkulepitud praktikatest ning headest tavadest pole veebilehtede elementide paigutuse kohta ühte kindlat standardit. Küll aga on tehtud mitmeid uuringuid, millised paigutused on kasutajate jaoks meeldivamad [80]. Suurettevõtted nagu Facebook, Google ja Twitter on autori arvates üritanud oma veebilehti kujundada

selliselt, et viimaste faktoritega oleks arvestatud. Autor otsustas kasutada oma klientrakenduse kasutajaliidese loomisel joonisel 18 välja toodud paigutust.



Joonis 18. Klientrakenduse kasutajaliidese paigutus

Selline elementide paigutus sarnaneb väga Google ning Facebooki veebilehtede paigutusele, mistõttu on see mitmetele õpilastele juba varasemast internetikasutusest tuttav.

Järgmise sammuna oli vajalik kasutajaliidese värvivaliku tegemine. Kasutajaliideses kuvatud värvide valik võiks tugineda värvide kokkusobivuse teooriale ning peaks vastama WCAG ehk *Web Content Accessibility Guidelines* standardile. Autor otsustas oma klientrakenduse kasutajaliidese värviteema valikul tugineda Politsei- ja Piirivalveameti stiilijuhile, mis viimasele standardile vastab [81].

4.2.4 Klientrakenduse funktsionaalsuse arendus

Funktsionaalsuse arendamisel otsustas autor alustada kasutaja registreerimise ning sisselogimise protsessist. Mõlemad funktsionaalsused langevad autoriseerimise kategooriasse ning seetõttu oli see hea valida esimeseks etapiks, et järgnevatel sammudel see funktsionaalsus olemas oleks ning arendustööd ei peataks. Autoriseerimise funktsionaalsuse arendamise käigus selgus, et arendusprotsessi teeb tülikaks serverirakenduse poolt tagastatud *token*'ite 10-minutiline aegumine. Autor otsustas antud probleemi lahendamiseks serverirakenduse konfiguratsiooni muuta, et

need aeguksid iga 24-tunni järel. See konfiguratsioonimuudatus oli vaid arendusprotsessi ajaks, ning hiljem seadistati konfiguratsioon tagasi vanasse olekusse.

Järgmise sammuna otsustas autor analüüsi käigus välja selgitatud õpetajate funktsionaalsetele ning mittefunktsionaalsetele nõuetele luua vastava klientrakenduse loogika. Alustati õpetajate küsimuste panga loomise, muutmise ning kustutamise loogika loomisega. Selle käigus loodi õpetajate navigeerimise menüüsse valik „Küsimuste pank“, millele navigeerides avaneb veebilehe sisuosas uus vaade, kus kuvatakse sisse logitud kasutaja loodud küsimustikke. Sellesse vaatesse loodi nupp „Loo uus küsimustik“, mille avamisel asendub sisuosa uue küsimustiku loomise vormiga. Antud vormis tuleb õpetajal täita küsimustiku nimi, nähtavus, ning lisada küsimusi. Igale küsimusele on vajalik määrata vastuse tüüp, ning viimasest sõltuvalt üksikvastus või mitmed vastused. Mitme vastuse puhul tuleb õpetajal valida, millised neist on õiged. Küsimustiku salvestamiseks loodi nupp „Salvesta küsimustik“. Küsimustike muutmise ning kustutamise loogika suuri lisaarendusi ei nõudnud, selleks loodi vaid kaks nuppu „Muuda“ ning „Kustuta“, mis iga küsimustiku elemendil küljes on. Küsimustiku muutmise vorm kasutab sama komponenti, mida küsimustiku loomise vorm, erinevus on vaid selles, et vormi laadimisel laaditakse andmed serverirakendusest.

Eelmisele funktsionaalsusele sarnaselt loodi õpetajarollis olevale kasutajale ka mängude loomise, muutmise ning kustutamise funktsionaalsus. Navigeerimise menüüsse loodi valik „Minu mängud“, millele vajutades kuvatakse kasutajale tema loodud mängude loetelu. Sellesse vaatesse loodi nupp „Loo uus mäng“, mis asendab loetelu vaate uue mängu loomise vormiga. Mängu loomise vormi kuvamiseks peab kasutaja esmalt valima loodava mängu tüübi. Antud diplomitöö käigus lõi autor vaid ühe mängu tüübi, milleks on mälukaardid. Mängu tüübi valimise järgselt kuvatakse kasutajale vorm kaheteistkümne konfiguratsioonielemendiga. Iga element muudab loodava mängu konfiguratsiooni, ning igale valikule lisati ka selgitav tekst. Sarnaselt eelmisele funktsionaalsusele on ka sellel funktsionaalsusel ära kasutatud asjaolu, et andmeobjekti (sellel puhul mäng) muutmise vaade kasutab sama komponenti mida loomise vaade, lihtsalt andmed laetakse serverirakendusest.

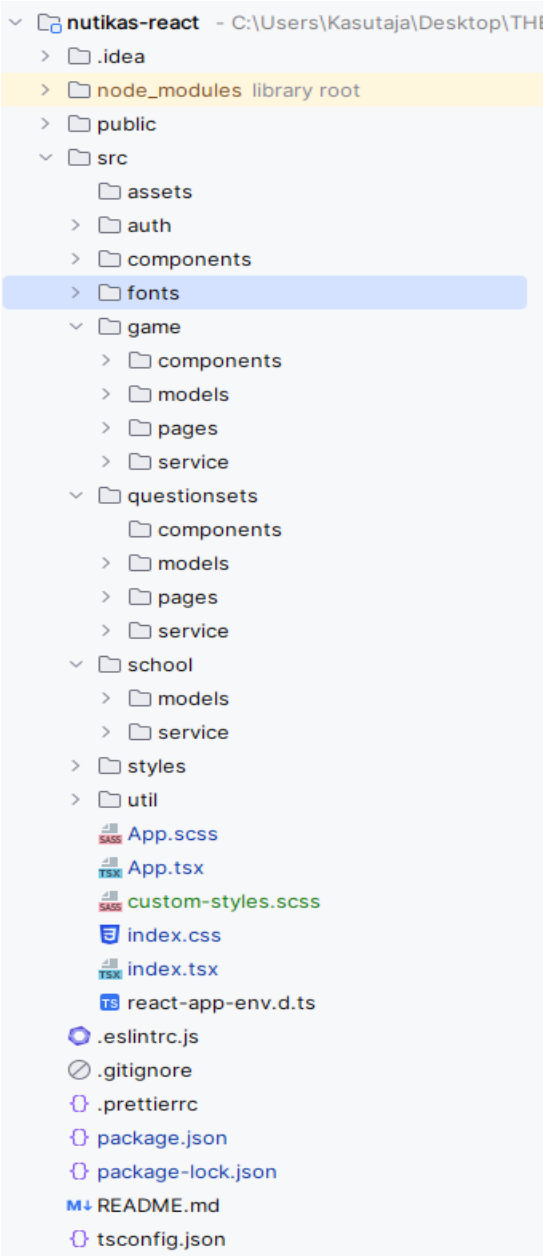
Õpetajate navigeerimise menüüsse loodi veel valikud „Edetabelid“, „Võistlused“, „Minu profiil“ ning „Seaded“, kuid antud diplomitöö käigus nendele valikutele funktsionaalsust ei loodud.

Järgmisena tuli luua klientrakendusse õpilaste funktsionaalsete ning mittefunktsionaalsete nõuete loogika. Alustati õpilastele mängude kuvamise loogikaga. Õpilase rollis oleva kasutaja vaatesse loodi navigeerimismenüüsse valik „Mänguväljak“, millele vajutades kuvatakse kasutajaliidese sisuosas mängude kogu. Mängud laetakse serverirakendusest õpilase kooli ja klassi põhjal, et kuvada vaid õpilasele asjakohased mängud. Samuti on selles vaates sisendkast „Mängu kood“ ning nupp „Liitu mänguga“, mis loob lisafunktsionaalsuse, kus õpetajad saavad jagada oma loodud mänge õpilastele, kes tegelikult nende klassi õpilaste alla ei kuulu. Mängu kood on seitsmekohaline ning koosneb ladina tähtedest ning araabia numbritest, see genereeritakse serverirakenduse poolt automaatselt. Mängule vajutades asendub mängude loetelu uue vaatega, kus õpilane saab vajutada nuppu „Alusta mängu“, mille puhul laaditakse serverirakendusest mängu parameetrid ning kuvatakse õpilasele mängu vaade, kus õpilane saab hakata mängima.

Õpilastele tasemete ning skoori kuvamiseks loodi kasutajaliidese üleval olevale ribale kaks uut elementi: õpilase taseme progressi näitav element ning tema kasutajaprofiili skoori arv. Neid elemente uuendatakse iga mängu lõppedes ning iga kord, kui klientrakendus serverist uuesti *token*’i pärib. Seda progressi haldab vaid serverirakendus, et vältida sohi tegemise võimalusi. Hetkel on progress vaid visuaalne, kuna tasemete ning skoori põhjal sisse logitud kasutaja kogemus ei muutu.

Õpilaste navigeerimise menüüsse loodi veel valikud „Edetabelid“, „Minu sõbrad“, „Võistlused“, „Minu profiil“ ning „Seaded“, kuid antud diplomitöö käigus nendele menüüelementidele funktsionaalsust ei loonud.

Eelnevalt välja toodud funktsionaalsuse arenduse käigus loodi mitmeid uusi React raamistiku komponente, mis jagati analüüsi käigus valitud kaustastruktuuri põhjal funktsionaalsuse järgi. Joonisel 19 on välja toodud klientrakenduse projekti struktuur peale arendusprotsessi.



Joonis 19. Loodud veebirakenduse struktuur

5 Tagasiside uuring

Tagasisideuuring viidi läbi samades fookusgruppides milles viidi läbi töö analüüsi etapis välja selgitatud õpilaste ja õpetajate nõuded ning soovid. Tagasisideuuringus osales 94 õpilast ning 13 õpetajat.

Tagasiside küsitlusele vastamise eelduseks oli loodud veebiplatvormi kasutamine vähemalt ühel korral. Töö autor andis õpilastele ning õpetajatele ligipääsu töö käigus loodud veebiplatvormile ning palus neil registreerida. Registreerimiseks vajalikud konfiguratsioonimuudatused tegi töö autor ise. Viimane tähendas serverirakenduse andmebaasi Tamsalu Gümnaasiumi kooli ning klasside andmeväljadesse informatsiooni lisamist. Testimise käigus loodi õpetajate poolt 36 küsimustikku ning 55 mängu. Kokku mängiti õpilaste poolt 50 erinevat õpetajate poolt mängu, mängitud mänge oli kokku 317.

Tagasiside küsitluse käigus kogus autor õpilaste ning õpetajate arvamusi kasutatud veebiplatvormi kohta vaba teksti vormis, punkthinnangutega ning jah / ei vastustega.

Küsimusele „Kui lihtne oli veebiplatvormi kasutada?“ skaalal 1-10 oli õpilaste vastuste keskmine hinnang 8,8 ning õpetajate keskmine hinnang 7,9. Õpetajad tõid välja mõned parendusettepanekud, näiteks oli kõige populaarsem parendusettepanek teha küsimustike vastuste lisamise protsess lihtsamaks. Töö autori arvamusel on antud punkthinnangud kõrged ning see viitab platvormi heale kasutatavusele ning lihtsusele, mis oli üks analüüsi käigus selgunud nõuetest.

Küsimusele „Kas tahaksid antud veebiplatvormi kasutada oma koolitöös?“ vastas jaatavalt 92% õpilastest ning 93% õpetajatest. Töö autori arvates kajastab see selle käesoleva töö asjakohasust ning sellise veebiplatvormi kasulikkust Eesti koolide haridusprotsessidele.

Küsimusele „Mis tekitas platvormi kasutamisel raskusi“ toodi välja mitu funktsionaalsust, mida antud veebiplatvormil pole: internetiühenduse puudumine, oma mängude ajaloo vaatamine, õpilaste progressi jälgimine. Kõik eelnimetatud funktsionaalsused on töö autoril plaanis tulevikus lisada.

Tagasiside põhjal on töö autoril plaanis teha parendusettepanekud ning neid testida.

6 Edasised plaanid

Töö autoril on käesolevas töös valmis arendatud veebiplatvormiga mitmeid sihte. Esiteks loodab töö autor luua veebiplatvormile veel mitmeid uusi funktsionaalsused: mitmed uued mängude tüübid, koolide ning õpilaste edetabelid, võistluste korraldamise loogika, küsimuste pankade jagamise loogika, õpilaste profiilide avatarid jpm. Kõik eelnimetatud funktsionaalsused on autori arvates olulised, et veebiplatvorm õpilastele ning õpetajatele veel meelepärasem oleks ning seda rohkem kasutataks.

Teiseks loodab töö autor anda veebiplatvormi võimalikult varakult kasutusse võimalikult paljudesse koolidesse. Veebiplatvormi aktiivselt kasutavate koolidega suhtlemine on autori arvates kõige parem viis koguda tagasisidet, et selle käigus valmiv toode oleks sihtgrupile meelepärane.

Kolmandaks loodab töö autor tulevikus teha koostööd Haridusministeeriumiga, et antud platvorm oleks tasuta kättesaadav kõigile seda soovivatele Eesti koolidele. Tuleviku vaates ei ole kindlasti mõistlik ega turvaline, et koolid kasutavad platvormi, mille töös hoidmiseks autoril vahendid puuduvad. Töö autor on juba ühendust võtnud Haridusministeeriumi töötajatega, et loodud veebiplatvormi neile tutvustada ning analüüsida võimalusi selle veebiplatvormi kättesaadavaks tegemiseks erinevatele Eesti koolidele.

Töö autor on arvamusel, et antud teema on aktuaalne ning selle teemaga kaasneb otsene kasu Eesti haridussüsteemile, mistõttu on autoril mõttes selles diplomitöös käsitletud probleemi ning veebiplatvormi edasi analüüsida magistriõppes.

7 Kokkuvõte

Käesoleva töö eesmärk oli luua uus veebipõhine platvorm, mis lahendab hetkel Eesti põhikooliõpilaste ja -õpetajate poolt kasutatavate platvormide murekohad ning loob uut väärtust õppeprotsessi mitmekülgsemaks tegemisel. Töö skoobis oli minimaalselt toimiva lahenduse loomine, mille nõuded valiti Tamsalu Gümnaasiumi põhikooliõpilastele ning nende õpetajatele suunatud küsitluse vastuste analüüsimisel.

Töö käigus tutvustati kolme eksisteerivat lahendust ning nende puuduseid. Sellele järgnes analüüsi osa, milles tutvuti lähemalt eksisteerivate probleemidega ning õpilaste ja õpetajate nõuetega. Töös analüüsi käigus on samuti välja toodud kasutusele võetud tehnoloogiad ning valikute tegemise põhjalik käsitlus. Tutvustatud on loodava veebirakenduse arhitektuuri ning disaini koos põhjendustega.

Töö arendusprotsessi käigus loodud taga- ning klientrakenduse ning andmebaasi loomist on põhjalikult käsitletud neljandas peatükis. Välja on toodud kasutatud tehnoloogiad ning nende valikute põhjendused ja disainiotsused. Käsitletud on mõningaid probleeme, millega töö autor arendusprotsessi käigus kokku puutus ning nende probleemide lahendusprotsessid. Eraldi on välja toodud testimisprotsess. Arendusprotsessile järgnes tagasisideuuring, mille käigus töö autor valis Tamsalu Gümnaasiumi põhikooliõpilaste ja -õpetajate seast fookusgrupi, kelle peal loodud rakenduse vastavust nõuetele testiti. Kogutud tagasiside on välja toodud ning autor on tagasisidet analüüsinud ning kommenteerinud.
Töö viimases osas on välja toodud tulevikuplaanid seoses valminud veebiplatvormi edasiarendusega ning Eesti koolidega.

Töö autor on arvamusel, et töö käigus valminud uus veebiplatvorm täidab töös seatud eesmärke ning lahendab aktuaalseid probleeme. Tulevikuplaanide ning valminud veebiplatvormi põhjal teeb autor järelduse, et on tugevalt panustanud Eesti koolide õppeprotsessi arengusse.

Kasutatud kirjandus

[1] V. Rogalevitš, „Digivahendid appi: kuidas luua tasuta interaktiivset e-õppe keskkonda,“ 11 08 2017. [Võrgumaterjal]. Available: https://www.ituudised.ee/uudised/2017/08/11/digivahendid-appi-kuidas-luua-tasuta-interaktiivset-e-oppe-keskkonda. [Kasutatud 01 04 2023].

[2] H. W. J. S. Olivia Johnston, „Teenagers learn through play too: communicating high expectations through a playful learning approach,“ *The Australian Educational Researcher,* 2022.

[3] M. Nutisport, „Nutisport,“ MTÜ Nutisport, 2015. [Võrgumaterjal]. Available: https://www.nutisport.eu. [Kasutatud 01 04 2023].

[4] M. Sõmer, „Nutispordi hinnakiri 2022/2023 (EURODES),“ [Võrgumaterjal]. Available: https://nutisport.eu/. [Kasutatud 13 04 2023].

[5] „What is Kahoot?,“ Kahoot!, 2023. [Võrgumaterjal]. Available: https://kahoot.com/what-is-kahoot/. [Kasutatud 01 04 2023].

[6] Kahoot!, „Choose the right Kahoot! 360 individual plan for you,“ [Võrgumaterjal]. Available: https://kahoot.com/business/pricing/. [Kasutatud 13 04 2023].

[7] R. Ellermaa, „Ellermaa tarkvaratöökoda,“ Ellermaa Tarkvaratöökoda OÜ, 2005. [Võrgumaterjal]. Available: http://www.ellermaasoft.ee/aa2.php. [Kasutatud 01 04 2023].

[8] Agile Business Consortium, „MoSCoW Prioritisation,“ Agile Business Consortium Limited, 2023. [Võrgumaterjal]. [Kasutatud 01 04 2023].

[9] J. S. D. J. R. E. C. B. A.-C. J. E. T. Hamari, „Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning,“ *Computers in Human Behavior,* pp. 54, 170-179, 2016.

[10] M. Rehkopf, „User stories with examples and a template,“ 2023. [Võrgumaterjal]. Available: https://www.atlassian.com/agile/project-management/user-stories. [Kasutatud 02 04 2023].

[11] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional, 2014.

[12] V. Puzhevich, „5 Steps for Choosing a Technology Stack for Your Project,“ 10 05 2022. [Võrgumaterjal]. Available: https://scand.com/company/blog/choosing-a-technology-stack/. [Kasutatud 02 04 2023].

[13] Internet Engineering Task Force, „Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing,“ 06 2014. [Võrgumaterjal]. Available: https://datatracker.ietf.org/doc/html/rfc7230. [Kasutatud 02 04 2023].

[14] RestApiTutorial, „Using HTTP Methods for RESTful Services,“ 2019. [Võrgumaterjal]. Available: https://www.restapitutorial.com/lessons/httpmethods.html. [Kasutatud 03 04 2023].

[15] PubNub, „7 Alternatives to REST APIs,“ 28 03 2023. [Võrgumaterjal]. Available: https://www.pubnub.com/blog/7-alternatives-to-rest-apis/. [Kasutatud 03 04 2023].

[16] Stackoverflow, „2022 Developer Survey,“ 2022. [Võrgumaterjal]. Available:

https://survey.stackoverflow.co/2022. [Kasutatud 04 04 2023].

[17] R. Winter, „JavaScript For Backend Development: An Introduction,“ HubSpot, 04 01 2023. [Võrgumaterjal]. Available: https://blog.hubspot.com/website/java-backend. [Kasutatud 05 04 2023].

[18] D. Bonderud, „The Beginner's Guide to Python Back-End Development,“ HubSpot, 13 12 2022. [Võrgumaterjal]. Available: https://blog.hubspot.com/website/python-backend. [Kasutatud 05 04 2023].

[19] G. v. Rossum, „Python 3 Reference Manual,“ CreateSpace, 2009. [Võrgumaterjal]. Available: https://docs.python.org/3/reference/. [Kasutatud 05 04 2023].

[20] H. Austerlitz, „Java,“ %1 *Data Acquisition Techniques using PCs (Second Edition)*, New York, Parker Hannifin Corp., 2003, p. 326.

[21] Microsoft, „A tour of the C# language,“ 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 05 04 2023].

[22] Kinsta, „PHP Market Share in 2023,“ 2023. [Võrgumaterjal]. Available: https://kinsta.com/php-market-share/. [Kasutatud 05 04 2023].

[23] Kinsta, „What Is a Content Management System (CMS)?,“ 20 03 2023. [Võrgumaterjal]. Available: https://kinsta.com/knowledgebase/content-management-system/. [Kasutatud 06 04 2023].

[24] V. M. R., „Top 10 Java Frameworks You Should Know,“ 29 11 2022. [Võrgumaterjal]. Available: https://www.edureka.co/blog/java-frameworks/. [Kasutatud 06 04 2023].

[25] Django Software Foundation, „https://www.djangoproject.com/,“ Django Software Foundation, 2023. [Võrgumaterjal]. Available: https://www.djangoproject.com/. [Kasutatud 06 04 2023].

[26] K. Mysliwiec, „Hello, nest!,“ Trilon, 2023. [Võrgumaterjal]. Available: https://nestjs.com/. [Kasutatud 06 04 2023].

[27] J. Almeida, „The 10 Best PHP Frameworks for Web Development in 2023,“ 1 02 2023. [Võrgumaterjal]. Available: https://distantjob.com/blog/best-php-frameworks-web-development/. [Kasutatud 06 04 2023].

[28] I. Gaba, „What is Gradle? Why Do We Use Gradle? Explained,“ 24 02 2023. [Võrgumaterjal]. Available: https://www.simplilearn.com/tutorials/gradle-tutorial/what-is-gradle. [Kasutatud 10 04 2023].

[29] Apache, „What is Maven,“ 19 04 2023. [Võrgumaterjal]. Available: https://maven.apache.org/what-is-maven.html. [Kasutatud 10 04 2023].

[30] N. Raval, „Top 10 JavaScript Usage Statistics to Prove Its Awesomeness in 2023,“ 03 04 2023. [Võrgumaterjal]. Available: https://radixweb.com/blog/top-javascript-usage-statistics. [Kasutatud 07 04 2023].

[31] N. Abramowski, „So, is HTML a Programming Language?,“ 03 06 2022. [Võrgumaterjal]. Available: https://careerfoundry.com/en/blog/web-development/is-html-a-programming-language/. [Kasutatud 07 04 2023].

[32] M. Joshi, „Angular vs React vs Vue: Core Differences,“ 23 12 2022. [Võrgumaterjal]. Available: https://www.browserstack.com/guide/angular-vs-react-vs-vue. [Kasutatud 07 04 2023].

[33] InsightSoftware, „What’s the Difference? Relational vs Non-Relational

Databases,“ 15 02 2021. [Võrgumaterjal]. Available: https://insightsoftware.com/blog/whats-the-difference-relational-vs-non-relational-databases/. [Kasutatud 07 04 2023].

[34] S. M. P. R. Carlos Coronel, „Types of Databases,“ %1 *Database Systems: Design, Implementation and Management*, Course Technology , 2010, p. 724.

[35] N. Osegi, „GOptimaEmbed: A Smart SMS-SQL Database Management System for Low-Cost Microcontrollers,“ National Open University of Nigeria, Lagos, 2015.

[36] DatabaseTown, „Relational Database Benefits and Limitations (Advantages & Disadvantages),“ 2023. [Võrgumaterjal]. Available: https://databasetown.com/relational-database-benefits-and-limitations/. [Kasutatud 08 04 2023].

[37] P. Raspel, „Relatsiooniline andmemudel,“ [Võrgumaterjal]. Available: https://enos.itcollege.ee/~priit/1.%20Andmebaasid/1.%20Loengumaterjalid/06/66988726001_6.htm. [Kasutatud 08 04 2023].

[38] C. C. Peter Rob, „The Relational Revolution,“ %1 *Database Systems: Design, Implementation and Management*, Course Technology, 2010, p. 33.

[39] Statista, „Ranking of the most popular relational database management systems worldwide, as of January 2022,“ 23 05 2022. [Võrgumaterjal]. Available: https://www.statista.com/statistics/1131568/worldwide-popularity-ranking-relational-database-management-systems/. [Kasutatud 08 04 2023].

[40] PostgreSQL Global Development Group, „PostgreSQL: The World's Most Advanced Open Source Relational Database,“ 2023. [Võrgumaterjal]. Available: https://www.postgresql.org/. [Kasutatud 08 04 2023].

[41] SQLite, „Most Widely Deployed and Used Database Engine,“ 08 01 2022. [Võrgumaterjal]. Available: https://sqlite.org/mostdeployed.html. [Kasutatud 09 04 2023].

[42] J. Momin, „Choosing the Right Database for Your Spring Boot Application,“ 27 02 2023. [Võrgumaterjal]. Available: https://www.linkedin.com/pulse/choosing-right-database-your-spring-boot-application-jahid-momin. [Kasutatud 09 04 2023].

[43] Umbraco, „What is a development environment?,“ [Võrgumaterjal]. Available: https://umbraco.com/knowledge-base/development-environment/. [Kasutatud 09 04 2023].

[44] Atlassian, „What is version control?,“ [Võrgumaterjal]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control. [Kasutatud 10 04 2023].

[45] Sonatype, „What is a code repository?,“ [Võrgumaterjal]. Available: https://www.sonatype.com/launchpad/what-are-code-repositories. [Kasutatud 10 04 2023].

[46] Amazon AWS, „What Is An IDE (Integrated Development Environment)?,“ [Võrgumaterjal]. Available: https://aws.amazon.com/what-is/ide/. [Kasutatud 10 04 2023].

[47] N. Stickman, „SVN vs Git: Which Version Control System Should You Use?,“ 09 03 2023. [Võrgumaterjal]. Available: https://www.linode.com/docs/guides/svn-vs-git/. [Kasutatud 10 04 2023].

[48] Atlassian, „Bitbucket: What is Bitbucket?,“ [Võrgumaterjal]. Available:

https://confluence.atlassian.com/confeval/development-tools-evaluator-resources/bitbucket/bitbucket-what-is-bitbucket. [Kasutatud 10 04 2023].

[49] Atlassian, „Compare plans and pricing,“ [Võrgumaterjal]. Available: https://www.atlassian.com/software/bitbucket/pricing. [Kasutatud 10 04 2023].

[50] Kinsta, „What Is GitHub? A Beginner’s Introduction to GitHub,“ 13 12 2022. [Võrgumaterjal]. Available: https://kinsta.com/knowledgebase/what-is-github/. [Kasutatud 11 04 2023].

[51] Jetbrains, „IntelliJ IDEA - the Leading Java and Kotlin IDE,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/idea/. [Kasutatud 11 04 2023].

[52] Microsoft, „Code editing. Redefined.,“ [Võrgumaterjal]. Available: https://code.visualstudio.com/. [Kasutatud 11 04 2023].

[53] IBM, „What is three-tier architecture?,“ [Võrgumaterjal]. Available: https://www.ibm.com/topics/three-tier-architecture. [Kasutatud 11 04 2023].

[54] A. Patil, „3 Layered Architecture,“ 09 05 2022. [Võrgumaterjal]. Available: https://www.ecanarys.com/Blogs/ArticleID/76/3-Layered-Architecture. [Kasutatud 10 04 2023].

[55] ClickIT, „Web Application Architecture: The Latest Guide 2022,“ [Võrgumaterjal]. Available: https://www.clickittech.com/devops/web-application-architecture/. [Kasutatud 11 04 2023].

[56] Gradle, „Organizing Gradle Projects,“ [Võrgumaterjal]. Available: https://docs.gradle.org/current/userguide/organizing_gradle_projects.html. [Kasutatud 10 04 2023].

[57] JavaTpoint, „Spring Boot Architecture,“ [Võrgumaterjal]. Available: https://www.javatpoint.com/spring-boot-architecture. [Kasutatud 10 04 2023].

[58] SAP, „Separation if Concerns,“ [Võrgumaterjal]. Available: https://help.sap.com/doc/abapdocu_753_index_htm/7.53/en-US/abenseperation_concerns_guidl.htm. [Kasutatud 10 04 2023].

[59] Facebook Open Source, „File Structure,“ 2023. [Võrgumaterjal]. Available: https://legacy.reactjs.org/docs/faq-structure.html. [Kasutatud 10 04 2023].

[60] J. Nielsen, „Usability 101: Introduction to Usability,“ 03 01 2012. [Võrgumaterjal]. Available: https://www.nngroup.com/articles/usability-101-introduction-to-usability/. [Kasutatud 11 04 2023].

[61] R. G. Acuña, „Colour your apps in React Native using Material Palette,“ 21 08 2017. [Võrgumaterjal]. Available: https://www.callstack.com/blog/colour-your-apps-in-react-native-using-material-palette. [Kasutatud 11 04 2023].

[62] P. Raspel, „Olemi-suhte diagramm ja andmebaaside loomine,“ [Võrgumaterjal]. Available: https://enos.itcollege.ee/~priit/1.%20Andmebaasid/1.%20Loengumaterjalid/04/4.htm. [Kasutatud 11 04 2023].

[63] P. Raspel, „ERD komponendid. ERD-ga seotud mõisted,“ [Võrgumaterjal]. Available: https://enos.itcollege.ee/~priit/1.%20Andmebaasid/1.%20Loengumaterjalid/04/4.2.htm. [Kasutatud 11 04 2023].

[64] N. Opinaldo, „Top 10 Free ER Diagram Tools in 2023,“ 07 04 2023. [Võrgumaterjal]. Available: https://gitmind.com/er-diagram-tool.html. [Kasutatud 11 04 2023].

[65] JavaTpoint, „Lombok Java,“ [Võrgumaterjal]. Available: https://www.javatpoint.com/lombok-java. [Kasutatud 12 04 2023].

[66] VMwave Tanzu, „Building a RESTful Web Service,“ [Võrgumaterjal]. Available: https://spring.io/guides/gs/rest-service/. [Kasutatud 12 04 2023].

[67] VMware Tanzu, „Securing a Web Application,“ [Võrgumaterjal]. Available: https://spring.io/guides/gs/securing-web/. [Kasutatud 12 04 2023].

[68] VMware Tanzu, „Acessing Relational Data using JDBC with Spring,“ [Võrgumaterjal]. Available: https://spring.io/guides/gs/relational-data-access/. [Kasutatud 12 04 2023].

[69] Baeldung, „Database Migrations with Flyway,“ 11 07 2022. [Võrgumaterjal]. Available: https://www.baeldung.com/database-migrations-with-flyway. [Kasutatud 12 04 2023].

[70] A. Kumar, „Web Application Folder Structure for Spring MVC Web Projects,“ 06 05 2014. [Võrgumaterjal]. Available: https://vitalflux.com/web-application-folder-structure-spring-mvc-web-projects/. [Kasutatud 12 04 2023].

[71] J. Lamb, „Session Tokens Vs. JWTs: Choosing Your Session Management Solution,“ 26 10 2022. [Võrgumaterjal]. Available: https://devops.com/session-tokens-vs-jwts-choosing-your-session-management-solution/Session Tokens Vs. JWTs: Choosing Your Session Management Solution. [Kasutatud 12 04 2023].

[72] J. Clarke, SQL Injection Attacks and Defense, Elsevier, 2009.

[73] Docker, „Docker Compose overview,“ [Võrgumaterjal]. Available: https://docs.docker.com/compose/. [Kasutatud 12 04 2023].

[74] DockerHub, „Postgres,“ [Võrgumaterjal]. Available: https://hub.docker.com/_/postgres. [Kasutatud 12 04 2023].

[75] Hibernate, „Hibernate. Everything data.,“ [Võrgumaterjal]. Available: https://hibernate.org/. [Kasutatud 12 04 2023].

[76] M. Tyson, „What is JPA? Introduction to Java persistence,“ 20 05 2022. [Võrgumaterjal]. Available: https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html. [Kasutatud 12 04 2023].

[77] BMC, „The Importance of SOLID Design Principles,“ 15 06 2020. [Võrgumaterjal]. Available: https://www.bmc.com/blogs/solid-design-principles/. [Kasutatud 12 04 2023].

[78] R. Pankaj, „Spring MVC @RequestMapping Annotation,“ 03 08 2022. [Võrgumaterjal]. Available: https://www.digitalocean.com/community/tutorials/spring-requestmapping-requestparam-pathvariable-example. [Kasutatud 12 04 2023].

[79] Baeldung, „Versioning a REST API,“ [Võrgumaterjal]. Available: https://www.baeldung.com/rest-versioning. [Kasutatud 12 04 2023].

[80] J. C. L. Z. S. D. Y. Renee Garett, „A Literature Review: Website Design and User Engagement,“ 06 07 2016. [Võrgumaterjal]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4974011/. [Kasutatud 16 04 2023].

[81] Politsei- ja Piirivalveamet, „Graafika,“ [Võrgumaterjal]. Available: https://www.politsei.ee/et/stiilijuht/graafika. [Kasutatud 18 04 2023].

[82] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[83] MySQL, „What is MySQL?,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. [Kasutatud 08 04 2023].

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

Mina, Maiko Metsalu

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Universaalne veebipõhine platvorm põhikooliõpilaste mänguliseks e-õppeks“, mille juhendaja on Meelis Antoi
   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.

19.04.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Töö käigus loodud andmemudeli olemid ning nende semantika

Tabeli nimi	Semantika
school	Tabel veebiplatvormi kasutatavate koolide informatsiooni talletamiseks.
class	Tabel veebiplatvormi kasutatavate koolide klasside informatsiooni talletamiseks.
users	Tabel registreeritud kasutajate informatsiooni talletamiseks.
roles	Tabel veebiplatvormi rollide info talletamiseks.
users_roles	Vahetabel kasutajate ning nende rollide seoste hoidmiseks.
question_set	Tabel veebiplatvormis loodavate küsimuste panga küsimustike talletamiseks.
question	Tabel küsimustikesse loodud küsimuste talletamiseks.
answer	Tabel loodud küsimuste vastuste talletamiseks.
game	Tabel loodud mängude talletamiseks.
game_participant	Tabel mängu mängijate talletamiseks.
live_game	Tabel käimasolevate mängude talletamiseks.

Lisa 3 – Klientrakenduse poolt tehtavad HTTP päringud

Päringu aadress	Päringu tüüp	Päringu eesmärk
/api/v1/questionset/{id}	GET	Pärida serverist kõik õpetaja küsimustikud.
/api/v1/questionset/create	POST	Luua uus küsimustik.
/api/v1/game/list/student	GET	Pärida serverist kõik mängud, mis on õpilasele mängitavad.
/api/v1/game/join?{id}	POST	Liituda mänguga.
/api/v1/game/create	POST	Luua uus mäng.
/api/v1/game/list	GET	Pärida serverist kõik õpetaja poolt loodud mängud.
/api/v1/class/list	GET	Pärida serverist kõik kooli klassid.
/api/v1/auth/login	POST	Sisselogimine ja *token'i* pärimine serverist.
/api/v1/auth/register	POST	Registreerimine.
/api/v1/auth/refresh	POST	Aegunud *token'i* värskendamine.
/api/v1/school/create	POST	Uue kooli loomine.
/ws/game/{uuid}	GET	Mängu mängimise protsessi infovahetuseks vajaliku kanali avamine.