TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

ÜHISE PROGRAMMEERIMISÜLESANNETE KOGU NING KOODI KIRJUTAMISE HARJUTAMISPLATVORMI ARENDAMINE

Bakalaureusetöö

Deniss Potapenko

Juhendaja: Ago Luberg,
tarkvarateaduse instituut, MSc

Tallinn 2018

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus ning kinnitan, et esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.
Kinnitan, et antud töö koostamisel olen kõikide teiste autorite seisukohtadele, probleemipüstitustele, kogutud arvandmetele jmt viidanud.

Deniss Potapenko *(allkirjastatud digitaalselt)*

21.05.2018

ANNOTATSIOON

Antud töö raames luuakse aastatel 2015 – 2017 kirjutatud Java programmeerimise põhikursuse raames agregeeritud, unifitseeritud ja korrastatud ülesannete kogu, kuhu saab tulevikus loodavaid ülesandeid reeglite põhjal salvestada. Samuti on korrastatud ülesannetest kogutud statistikat, mis näitab, kuivõrd edukalt on antud ülesanne eelnevatel aastatel tudengite poolt lahendatud. Lisaks sisaldab ülesannete kogu endas reegleid, kuidas ülesandeid tuleks koostada ning kuidas neid automaattestidega katta.

Eesmärgi täimisel kasutatakse ülesannete hoidmiseks Gitlab salve, ülesannete kirjeldus kohendatakse ReStructuredText formaati ning testimisel kasutatakse TestNG raamistikku ja StudentTester moodulit.

Lisaks luuakse koodi kirjutamise harjutamisplatvorm, mis võimaldab välja pandud ülesandeid otse veebirakenduses lahendada kirjutades lähtekoodi ja automaattestide abil koheselt tagasisidet saada. Suur osa platvormil esindatud ülesannetest on võetud lõputöö raames loodud ülesannete kogust.

Harjutusplatvormi loomisel kasutatakse Java ning JavaScripti programmeerimiskeeli, Spring Boot ja Vue js raamistikke, MySQL andmebaasi, Vuetify komponendiraamistikku ja Codemirror tekstiredaktorit. Lisaks kasutatakse Jenkinsit ning Dockerit, et evitust automatiseerida ja rakendus konteineriseerida.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 3 peatükki ja 15 joonist.

ABSTRACT

First purpose of this thesis is to create aggregated, standardised and structured collection of programming exercises from years 2015 - 2017 “Java computer programming” course with the possibility to add new exercises to that collection. Furthermore, analysis of 2015 – 2017 exercises makes it possible to gather statistical information about each exercise that is also provided in this thesis. Statistics mostly cover average percentage of success within each exercise. Collection of exercises has a list of rules about how to compose new exercises for the collection and how to cover them with automatic tests.

Gitlab repository is used to store exercises. ReStructuredText formatter is used for structuring exercise texts. TestNG testing framework is used to write automatic tests and StudentTester is used for detailed tester output.

Second purpose of this thesis is to create web platform that allows students to solve programming exercises in the web browser and get the feedback immediately. Feedback is based on automatic tests. Most of the exercises present in the platform are taken from the collection of exercises that is created in the first part of this thesis.

Technologically Java and JavaScript were used as programming languages. Spring Boot and Vue js were used as frameworks. MySQL was used as database. Vuetify js was used as component framework and Codemirror was used as text editor. Jenkins and Docker were used to automate deployment process and containerize the application.

The thesis is in Estonian and contains 44 pages of text, 3 chapters and 15 figures.

SISUKORD

JOONISTE LOETELU

KASUTATUD LÜHENDID

API	*Application Programming Interface*, programmi liides
Backend	Serveripoolne arhitektuuri osa, mille tegevusaladeks on äriloogika ja andmed
DOM	*Document Object Model* Dokumendi objektimudel, mis on eeskirjaks selle kohta, kuidas objekte veebilehel esitada
Endpoint	Aadress, mille kaudu on võimalik veebiteenusega suhelda
Frontend	Kliendipoolne osa arhitektuurist, mis suhtleb serveriga või väliste teenustega
Git	Versioonihaldussüsteem
GUI	*Graphical User Interface* Graafiline kasutajaliides, mis võimaldab visuaalselt programmi manipuleerida erinevalt tekstipõhisest kasutajaliidesest.
HTTP	*Hypertext Transfer Protocol* Protokoll teabe edastamiseks arvutivõrkudes
HTTP GET	Idempotentne HTTP päringu tüüp, REST puhul kasutatakse andmete küsimiseks
HTTP POST	Idempotentne HTTP päringu tüüp, REST puhul kasutatakse andmete lisamiseks
IDE	*Integrated Development Environment* Integreeritud arenduskeskkond, mille eesmärgiks on tarkvaraarenduse lihtsustamine.
JAR	*Java Archive* Pakendamise formaat, kus on agregeeritud Java klassifailid, *metadata* ja ressursid.
JSON	*Javascript Object Notation*, Andmeedastuse tarvis loodud tekstiformaat
NPM	*Node Packer Manager* Suurim ühtne Javascripti pakettide salv
ORM	*Object-relational mapping* Virtuaalne objektide andmebaas mis ühendab andmebaasi ja serverirakenduse
URI	*Uniform Resource Identifier* Sümbolite jada, mis märgistab loogilist või füüsilist ressurssi

SISSEJUHATUS

Aastate jooksul on Tehnikaülikooli programmerimise ainetes tudengite jaoks loodud hulgaliselt semestri- ja eksamiülesandeid eesmärgiga, et iga aasta saaks õpilased uusi ülesandeid lahendada. Ülesannete loomisesse on panustatud rohkelt aega ning suuremat osa neist oleks võimalik taaskasutada. Paraku ei ole hetkel olemas ühist salve, kus kõik varasemad ülesanded oleks kättesaadavad.

Antud töö esimeseks eesmärgiks on luua ühine agregeeritud ülesannete kogu, kuhu on salvestatud varasemate aastate ülesanded koos nendega seotud automaattestidega, mida oleks võimalik vajadusel tudengitele lahendamiseks anda.

Teine lõputöö eesmärk on programmeerimisülesannete harjutusplatvormi loomine, mis aitaks tudengitel oma koodikirjutamise oskusi parandada harjutamise käigus. Kuigi sarnaseid platvorme on viimasel ajal loodud hulgaliselt, siis on tihti nende puhul probleem, kui soovitakse oma loodud ülesanded üles laadida ning tudengitele anda. TTÜ tarvis loodud platvorm võimaldaks välja panna eelnevatel aastatel kasutatud ülesandeid, millest osa on loodud antud lõputöö esimeses osas, et tudengid saaks koodi kirjutamist harjutada.

Töö alaosadeks jaotus on tehtud järgnevalt: taustauuringus kirjeldatakse hetkel turul olevaid programmeerimise õppe- ja harjutusplatvorme ning võrreldakse neid omavahel. Samuti uuritakse ning tuuakse välja, kuidas hetkel TTÜ-s programmeerimise aineid korraldatakse ning ülesandeid koostatakse. Järgnevas peatükis kirjeldatakse programmeerimisülesannete kogu koostamist, selle kohta käivate nõuete esitamist, arendusmetoodikat ja protsessi ning loodud funktsionaalsust. Seejärel selgitatakse harjutamisplatvormi loomist, selle analüüsi, arendusprotsessi ja metoodikaid ning lõpuks loodud funktsionaalsust ja projekti struktuuri. Kokkuvõttes võetakse tulemused sissejuhatuses sõnastatud eesmärkide kaupa kokku ning tuuakse välja töö käigus välja tulnud järeldused.

1. TAUSTAUURING

1.1. Programmeerimise õppe- ja harjutamisplatvormid

Viimase 20 aasta jooksul on nõudlus infotehnoloogia järel olnud pidevalt tõusuteel ning järgnevatel aastatel ei ole põhjust arvata, et positiivne trend peaks lõppema seoses selliste tehnoloogiate nagu virtuaalne reaalsus, asjade internet ja mikroteenuste esile kerkimisega. [1], [2]

Seoses valdkonna populaarsusega on ka koodi kirjutamine muutunud viimasel ajal kättesaadavamaks. Ülikoolid pakuvad kaugõppe programme arvutiteaduse valdkonnas ning on laialdaselt levinud programmeerimisringid kooli õpilastele. Informaatika tundides on tänapäeval palju rohkem kokkupuudet koodi kirjutamisega, kui seda oli kümmekond aastat tagasi. Käivitatakse pilootprogramme, kus lapsi õpetatakse alates esimesest klassist programme koostama. [3]

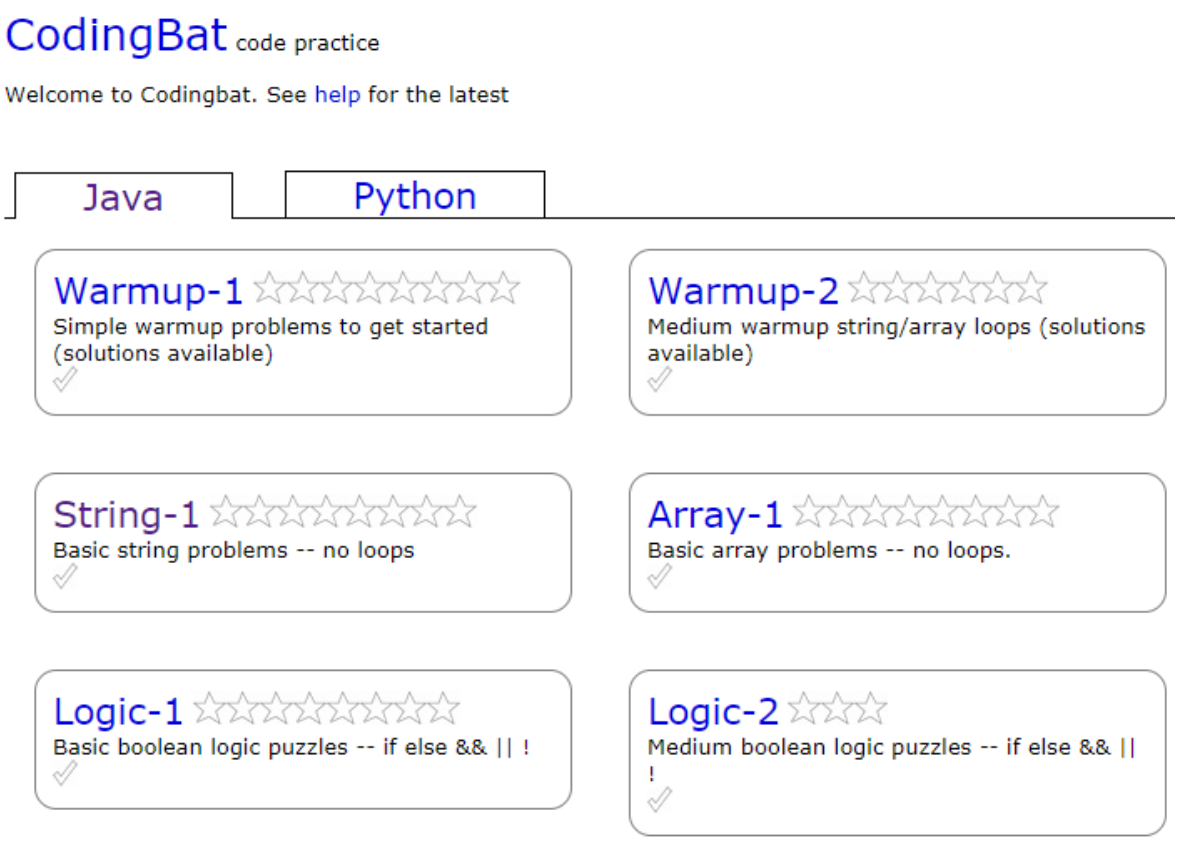
Viimaste aastate trendina on esile kerkinud ka koodi kirjutamise veebiplatvormid, millel on küllaltki erinevad eesmärgid, kuid ühine põhimõte. Nende õppeplatvormide abil on tehtud koodi kirjutamine võimalikult lihtsaks ja mugavaks. Käivituskeskkonna ülesseadmine on veebiplatvormi looja poolt sätitud ning kasutaja ei pea kompilaatorite allalaadimisele, IDE-de seadistamisele ning automaattestide kirjutamisele aega kulutama. Kasutajad saavad programmeerimisele keskenduda.

Koodkirjutamise õpetamise platvormidest on üks tuntuimatest codecademy [1], kuhu on end registreerinud üle 45 miljoni inimese. Selle veebiplatvormi eesmärgiks on algtasemel koodi kirjutamise õpetamine populaarseimates keeltes. Kõige suuremaks eeliseks on samm-sammulised juhendind, kus on kõrvuti teoreetilised alused ning koodinäited. Õpilane peab iga peatüki juures lahendama kindlaid ülesandeid, mille korrektsust kontrollitakse automaattestide abil.

[1] Codecademy veebileht. [WWW] https://www.codecademy.com

Lihtsamatest platvormidest, millest antud lõputöö on osaliselt inspireeritud, on Codingbat [2]. Vaatamata oma lihtsale disainilahendusele on selle veebiplatvormi eeliseks suur hulk eri raskusastmega ning erinevatesse kategooriatesse kuuluvaid ülesandeid.

Selle platvormi eesmärgiks on tehniliste oskuste arendamine eri tüüpi ülesannete lahendamisel. Puuduseks võib pidada vananenud disainilahendust ning ainult kahe programmeerimiskeele, Pythoni ning Java, toetamist. Samuti ei ole Codingbatis võimalik objektorienteeritult ülesandeid lahendada, kuna platvorm ei toeta mitme klassi loomist. Platvormi kasutajaliides on illustreeritud joonisel 1.



Joonis 1 Ülesannete valik Codingbat keskkonnas

Eraldi vajab mainimist üks vähestest suurtest koodi kirjutamise platvormidest, millel on avatud lähtekood ning millest antud lõputöö sai arhitektuurilisi mõtteid. Exercismi [3] koduleheküljel mainitakse, et ta on mõeldud algajate jaoks, kes saavad algtasemel teadmisi omandada, või edasijõudnud programmeerijate jaoks, kes soovivad uute keeltega tuttavaks

[2] Codingbat veebileht. [WWW] https://www.codingbat.com
[3] Exercism veebileht [WWW] http://exercism.io/

saada. Teadaolevalt saab Excercismis lahendada ülesandeid enam kui 40 keeles. Suurimaks mureks on kasutajakogemus, mille üle kurdab ka projekti eestvedaja [4]. Nimelt ülesannete lahendamisel ei ole pakkuda brauseripõhist koodi kirjutamise akent, vaid õpilane peab tagasiside saamseks igat ülesannet Giti abil alla kloonima ning lahenduse Giti üles panema. Selline protsess on algaja jaoks ebaotstarbekalt keeruline.

Lähtekoodi uurides võib tähele panna, et tegemist on Ruby keeles kirjutatud projektiga. Samaaegselt on kasutajaliideses kasutatud raamistikest Angulari kui ka puhast JavaScripti ja CoffeeScripti. Samuti on projekti tarvis tehtud eraldiseisev suur tester, mis oskab 40 keeles lahendatud ülesandeid testida. Exercism platvormis ei ole tegemist sisend/väljund testidega ning igas keeles on ülesanded erinevad. See on võimalik, kuna tegemist on avatud lähtekoodiga projektiga, kus inimesed saavad oma ülesandeid ja teste üles panna. Sarnaselt antud lõputööga on Exercism võtnud kasutusele pidevtarne tarkvarana Jenkinsi ja hoiab oma projekti Docker konteineris. Mõlemas projektis on ülesannete hoidmiseks kasutatud relatsioonilist andmebaasi. Exercismi puhul on andmebaasiks Postgresql ning antu lõputöö puhul MySQL. Samas on lõputöö raames integreeritud koodi kirjutamise aken otse brauserisse, selleks, et olla võimalikult algajasõbralik.

Kuigi kõikidel platvormidel on palju eeliseid ning kõiki kasutades oleks võimalik programmeerimise oskust parandada, siis kahjuks puudub suuremal osal nendest platvormidest võimalus lisada oma ülesandeid, mis on üheks nõudeks antud töös. Exercism võimaldab ülesandeid lisada, kuid grupipõhist ülesannete kategoriseerimist ei ole. Samuti oleks lõputöös loodava platvormi nõudeks võimalus tulevikus integreerida ta selliselt, et saaks sisse logida Moodle [4] kasutajanime ja parooliga ning lahendatud ülesandeid hinnata ja tudengile punkte anda. Sellist integratsiooni valmisproduktina eksisteerivad platvormid ei võimalda.

1.2. Ülesannete koostamisest TTÜ programmeerimisainetes

Antud peatükis on vaatluse alla võetud ülesannete koostamine programmeerimise algkursuse ja põhikursuse ainetes.

Kuna ülikooli kursuste planeerimisel tuleb silmas pidada nende lühikest kestust (16 nädalat), siis õppenädalate jooksul peab tudengitele tegema tutvustust suure teemade hulgaga. Seoses

[4] Moodle TTÜ leht [WWW] https://ained.ttu.ee

sellega on iganädalased ülesanded keskendunud erisugustele ülesannete tüüpidele, kus Python algkursuse esimesed nädalad keskenduvad andmetüüpide manipuleerimisele, funktsiooni mõistmisele, sobivate andmestruktuuride kasutamisele ning hiljem objekt-orienteeritusele ja keerukamadele teemadele nagu rekursioon ja teekide kasutamine.

Java põhikursus jälgib samuti sarnast põhimõtet, kuid OOP tuleb palju varem ning viimased ülesanded on seotud algoritmidega ja API-dega. Vahepeal tehakse ka graafilise kasutajaliidesega ülesandeid.

Suurem osa ülesandeid on kaetud automaattestidega, et õppejõul oleks lihtsam kontrollida ülesande loogika korrektsust. Algkursuse raames kasutatakse Unittest [5] raamistikku Pythoni ülesannetel ning põhikursuse raames kasutatakse TestNG [6] raamistikku ning automaattestimist hõlbustavat moodulit StudentTester. [7]

Iga aasta on nende kursuste raames loodud uusi ülesandeid selle mõttega, et tudengid ei esitaks oma nime alt varasemate tudengite ülesandeid. Pidevalt uued on nii semestri ülesanded kui ka eksamiülesanded.

Uute ülesannete pidev koostamine on loonud sellise olukorra, kus eelmiste aastate ülesanded jäävad taaskasutamata. Selle lõputöö üheks eesmärgiks on teha kasutuskõlbulikuks vanemad ülesanded jagades nad harjutusplatvormi ja tulevaste eksamite vahel.

1.3. Koodi kirjutamise platvormi raames kasutatud tehnoloogiad

Järgnevates peatükkides kirjeldatakse antud töö raames olulisemaid kasutatavaid tehnoloogiaid.

1.3.1. Spring boot

Springi [8] puhul on tegemist populaarseima Java raamistikuga ettevõtte tasemel rakenduste loomisel [5], mille moodulit Spring Booti selles lõputöös kasutatakse. Raamistik võimaldab teha valutuks infrastruktuuri haldamise Java rakendustes ning aitab arendajal keskenduda puhta Java koodi kirjutamisele. Spring pakub omapoolset lahendust rakendades mustreid

[5] unittest raamistik. [WWW] https://docs.Python.org/3/library/unittest.html.
[6] TestNG raamistik. [WWW] http://testng.org/doc/
[7] Andres Antonen. StudentTester. [WWW] https://github.com/ndrez93/StudentTester
[8] Spring veebileht. [WWW] https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/overview.html.

sellistele igapäevastele arendaja probleemidele nagu Java meetodi otsene suhtlus andmebaasiga, koodiväline konfigureeritavatus ning rakenduse tagasiulatuv sobilikkus.

Spring Boot on inspireeritud David Heinemeier Hanssoni ütlusest „*Convention over configuration*“, ehk kokkulepped enne konfigureeritavust. Selle mooduli puhul on võimalik väikseima vaevaga rakenduse ülesseadmine igas süsteemis ilma suurema konfiguratsioonita, mis on Spring raamistikus väga laialtlevinud. Spring Boot [9] pakub koheselt käivituvaid JAR faile, mida saab ka veebirakendustena vajadusel kasutada, kuna sinna sisse on ehitatud Apache Tomcat server. Spring Booti vaikeseadistuse nõrkuseks on väiksem võimalus koodi oma meele järgi konfigureerida. Vähem levinud olukorrad võivad nõuda arendajatelt iseseisvalt konfiguratsiooni failide tekitamist.

1.3.2. Hibernate

Hibernate ORM [10] on selle töö raames lahutamatu osa Spring Boot rakendusest. Üldiselt on tegemist virtuaalse objektide andmebaasiga, mis ühendab relatsioonilise andmebaasi ning Java rakenduse. Selle vabavaralise tehnoloogia abil on võimalik kaardistada andmebaasi tabelid Java klasside järgi ning seejärel teha andmebaasi päringuid otse Java koodist. Hibernate kasutamine Spring Boot rakenduses on väga lihtne ning selle konfigureerimine käib läbi ühe tekstifaili muutmise.

1.3.3. Vue js

Veebirakenduse kasutajaliidesena on antud lõputöö raames kasutatud Vue js raamistikku [11], mille abil on võimalik luua dünaamilisi ühelehelisi veebirakendusi. Vue eeliseks muude suurte JavaScripti raamistike kõrval on kiirus, dünaamilisus, arendajasõbralikkus, taaskasutatavus, laiendavatus ning kompaktsus, Vue js tuumraamistiku failisuuruseks on ainult 16kb. [6] Puuduseks on see, et võrreldes teiste suurte eeskomponentidega nagu React ja Angular on Vue puhul tegemist pigem uue raamistikuga, mille kohta ei ole veel niivõrd palju dokumentatsiooni ja teadmiste jagamist kasutajate vahel.

Arendajasõbralikkuse tagab see, et Vue arendamiseks piisab tavalisest JavaScripti ja HTMLi teadmisest. Kogu andmete sidumine virtuaalse DOMiga toimub läbi HTML *tag*’ide, mis on

[9] Spring boot veebileht [WWW] https://projects.spring.io/spring-boot/
[10] Hibernate veebileht [WWW] http://hibernate.org/orm/
[11] Vue js veebileht [WWW] https://vuejs.org/

oma olemuselt väga intuitiivne. Lisaks ei eelda Vue kasutamine hulgaliste JavaScripti teekide teadmist või Typescripti oskusi nagu on seda vaja Angulari programmeerimisel.

Taaskasutatavuse tagab komponentide süsteem, kus iga komponent on eraldiseisev tükk koodist või veebilehest, mida saab puhtal kujul taaskasutada või vajadusel teistest komponentidest lahus hoida. Komponentideks jagamine aitab veebirakenduse arenduse kiiremaks teha, kuna sõltumatud ja hästi piiritletud komponendid tagavad selle, et lähtekoodi muutmine ühes kohas ei tekita vigu teistes komponentides. [15]

Laiendatavuse tagab tuumikraamistiku väike failisuurus ja kiirus ning see, et vajadusel on võimalik enda jaoks sobivaid mooduleid lisada. Selle töö raames on kasutatud lisateegina vue-routerit, mis võimaldab komponentide kaardistamise vastavalt URI marsruutidele. Moodul Vuetify.js [12] annab ligipääsu *Material Design* komponentidele, mis teevad lihtsaks veebilehe kujundusaspekti.

1.3.4. MySQL

Ülesannete säilitamiseks kaustatakse antud töös relatsioonilist andmebaasi MySQL [13], mis on populaarsuselt teine andmebaasisüsteem. [7] MySQL eeliseks on vabavaralisus, kiirus ja kasutajasõbralikkus. MySQL ülesseadmine on kiire ning Spring Boot serveripoolega ühendamine on väga levinud ning hästi dokumenteeritud Spring rakenduste seas. Samuti on relatsioonilised andmebaasid kõige tuttavamad autori jaoks.

1.3.5. Docker

Dockeri [14] puhul on tegemist aina populaarsemaks muutuva minimaalse virtuaalmasinaga, mis võimaldab rakenduste konteineritesse panemist. Dockeri kodulehel on Docker konteineri mõistet defineerides öeldud, et konteiner on kerge, iseseisev, taaskäivitatav tarkvara pakett, mille sees on kõik, mis käivitamiseks vajalik on.

Dockeri eeliseks traditsiooniliste virtuaalmasinate ees on palju madalam arvutiressursside hõivamine, käivituskiirus, mis kestab sekundeid võrreldes virtuaalmasinate minutiliste käivitusfaasidega. Konteinerid lihtustavad tarkvara arenduse, testimise ja evituse protsessi ning Docker garanteerib, et jagatav konteiner töötab igas keskkonnas ühtemoodi. Dockeri kasutamisel ei ole vaja konfigureerida käivituskeskkonda, kuna see asub koos rakendusega

[12] Vuetify.js veebileht [WWW] https://vuetifyjs.com/en/
[13] MySQL koduleht [WWW] https://www.mysql.com/
[14] Docker dokumentatsioon [WWW] https://docs.docker.com/

konteineri sees. Konteinerid kasutavad samu ressursse ning seega piisab palju väiksematest *host* masina, ehk servermasina, kus Dockeri konteiner töötab, algressurssidest, et mitmeid konteinereid püsti hoida.

Juhul, kui masinasse on installeeritud Docker, siis võib avalikust Docker Hub salvest alla tõmmata sobiva konteineri ning see oma masinas kõikide sõltuvustega koheselt käima panna.

Veel üheks Dockeri eeliseks on ühine ja üksainus Dockerfail, mis määrab ära kogu programmi töö konfiguratsiooni. [8]

1.3.6. Jenkins

Jenkinsi [15] puhul on tegemist vabavaralise pidevtarne (*continous delivery*) tarkvaraga, mis võimaldab programmide ülesanded automatiseerida. Jenkins nõuab minimaalselt hooldust ning on intuitiivne tänu oma sisseehitatud GUIle. Tarkvara võimaldab automatiseeritud evitust, testimist, pakendamist ja analüüsi. Kõikide automatiseeritud tegevuste kohta on võimalik saada informatsiooni ning juhul, kui automaatne protsess ebaõnnestub, siis on vastavad teavitused kohe olemas. Avatud lähtekood võimaldab arendajate kogukonnal enda poolt kirjutatud pistikprogramme Jenkinsile pakkuda. Seetõttu on paljud spetsiifilised pidevtarne stsenaariumid kaetud.

Automatiseerimise järjekordseks eelisteks on väga suur ajaline kokkuhoid, kuna manuaalne inimesepoolne evituse protsess on tihtipeale väga aeglane ning veaohtlik.

[15] Jenkins veebileht [WWW] https://jenkins.io/

2. ÜHISE PROGRAMMEERIMISÜLESANNETE KOGU KOOSTAMINE

Järgnevalt on kirjeldatud programmeerimisülesannete agregeerimisel ja korrastamisel välja toodud nõuded. Seejärel on kirjeldatud kogu koostamise arendusprotsessi ning pärast seda on kirjeldatud ülesannete korrastamisel saavutatud tulemus. Lahti on selgitatud tekkinud salve struktuur, sisu ning kasutusjuhend. Samuti on välja toodud töö käigus koostatud statistika eksamitel olnud ülesannete lahendamise kohta.

2.1. Ülesannete kogu koostamise nõuded

Järgnevalt on loetletud tähtsamad nõuded, millega kogu koostamise käigus arvestatakse. Korrastatud ülesanded koosnevad kahest failist: tekstifail ning testifail

- Tekstifail sisaldab ülesande unikaalset pealkirja, raskusastet, kirjeldusteksti eesti keeles, kirjeldusteksti inglise keeles, sisend-väljund paare ning oodatavat funktsiooni nime koos parameetritega.
- Testifail koosneb häid koodi kirjutamise tavasid järgivatest testisammudest ehk testfunktsioonidest
- Igas testifailis on 3 automaattesti lihtsal ülesanded, 5 keskmisel ja 10 raskel
- Iga testifail sisaldab juhuteste
- Igale ülesandele on kõrvale panna ka lahenduse lähtekood, mille abil saab ülesannet edukalt lahendada.
- Kogus on tabel kõikide ülesannete kohta käivate võtmesõnadega, mis võimaldab sobivat ülesannet kiiresti üles otsida.
- Lisaks on tabelisse kogutud statistika, mis näitab, kuidas on antud ülesandega eelmistel aastatel tudengid hakkama saanud
- Kogus on olemas ka juhend, kuidas kogusse oma ülesandeid lisada ning milliseid põhimõtteid seejuures jälgida

2.2. Agregeerimise ning korrastamise arendusmetoodika ning protsess

Antud töö osas on eesmärgiks koostada ühine salv, kuhu oleks agregeeritud erinevate õppeaastate programmeerimise ülesanded. Arendusprotsess jaguneb kaheks:

Automaattestide kirjutamine ning korrastamine ja ülesannete tekstide kirjutamine, tõlkimine ning sõnastuse parandamine.

Selles töös tegeletakse põhilisest Java põhikursuse eksamiülesannetega, kuna lisaks ajalisele piirangule on selle töö kirjutamise ajal käimas kevadine „Programmeerimise põhikursus Javas“ ning korrastatud eksamiülesandeid saab eksamil kasutada. Töö käigus tekib struktuur, mida saab tulevikus kasutada ükskõik milliste ülesannete kategoriseerimisel.

2.2.1. Testide kirjutamine ja korrastamine

Kuna eksamiülesanded pärinevad aastatest 2015-2017, siis on kõikide ülesannete kuju, kasutatud testimise raamisik ja testide hulk erinev. Aasta 2017 ülesannete testid on kõige paremas seisus ning seepärast oli neid töö käigus vaja ainult korrastada. Korrastamine tähendab näite sisend-väljundite lisamine ja teksti kohendamine. Selleks, et 2015-2016 jooksul loodud ülesandeid ühtlustada, on kasutatud 2017 aastaga sarnaselt testimisraamistikuna TestNGd [16], millele on lisatud StudentTester moodul, mis võimaldab testide väljundit detailsemaks teha.

Kuna harilikult ei kata automaattestid kõiki sisendeid, siis antud töös on peaaegu igale (~ 90%)ülesandele proovitud teha juurde juhuteste, juhul kui neid algselt ei olnud, mis aitavad kaasa, et programm töötab õigesti iga sisendi korral.

Ülesannete ühtlustamisel on tehtud punktide järgi tasakaalustamist, mis tähendab seda, et näiteks keskmine ülesanne, mis annab 3 punkti eksamil, on testitav 3 testiga. Selline struktuur on suuremal osal ülesannetest.

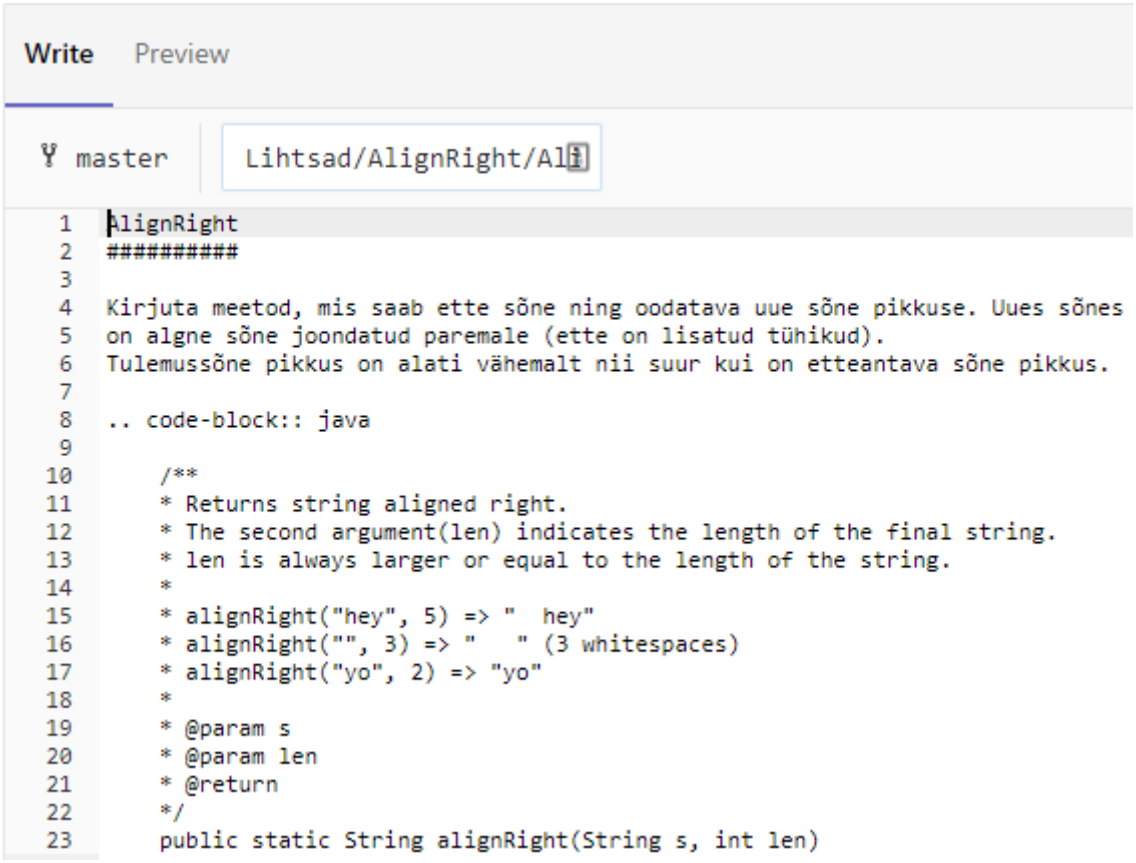
2.2.2. Tekstide korrastamine

Tekstide korrastamise etapis on läbi töötatud kõik tekstid ning parandatud keerulisemates kohtades sõnastust. Vajadusel on mõningatele ülesannetele lisatud selgitavaid illustratsioone. Kuna 2017 aasta ülesannete juures oli algselt kirjutatud inglisekeelne selgitav Javadoc, siis on antud töö käigus lisatud inglisekeelne dokumentatsioon kõikide ülesannete juurde, mis pärinevad varasematest aastatest. Samuti on lisatud hulgaliselt näidissisendeid ja väljundeid kohtadesse, kus neid ei olnud piisavalt.

[16] TestNG veebileht [WWW] http://testng.org/doc/documentation-main.html

Teksti kujunduseks on kasutatud reStructuredTexti [17] , mis on dokumentatsiooni kirjeldamiseks mõeldud struktureeritud faili formaat, mis võimaldab mugavalt tehnilist dokumentatsiooni kirjutada ning seda ka lugeda. Oma olemuselt sai valitud just see formaat, kuna eelnevalt oli sellega kogemusi programmeerimise ainetes dokumentatsiooni kirjutamisel.

Seda formaati kasutatakse ülesannete ning näitekoodi kirjeldamiseks ning konkreetne ülesande tekstifaili lähtekood on välja toodud joonisel 2.



Joonis 2 Näide .rst formaadis ülesande tekstist

2.2.3. Salves hoidmine

Kuna Programmeerimise algkursuse ja põhikursuse ülesanded, testid ning dokumentatsioon asuvad Gitlab versioonihaldussüsteemis, siis asetseb antud töö käigus loodud agregeeritud ülesannete kogu järjepidevuse ja sarnase eesmärgi tõttu samuti Gitlabis.

[17] reStructuredText veebileht [WWW] http://docutils.sourceforge.net/rst.html

Agregeeritud kogu puhul on tegemist eelkõige lähtekoodiga ülesannete juurde kuuluvate testide näol. Tekstide kirjutamisel on samuti kasutatud struktureeritud teksti formaati. Seetõttu on väga mugav kirjutada teste ja teksti IDEs ning seejärel versioonihaldussüsteemiga sünkroniseerida.

2.3. Salve struktuur ja funktsionaalsus

Ülesannete agregeerimise ja korrastamise käigus on loodud Gitlab salv [18], kuhu on koondatud aastate 2015 – 2017 „Programmeerimise põhikursuse Javas“ raames kasutatud eksamiülesanded. Ülesanded on jaotatud raskusastme järgi kolme kategooriasse:

- Lihtsad on ülesanded, mida keskmine tudeng lahendab kuni 10 minutiga ning kuni 10 rea koodiga. Kõige enam iseloomustab neid üks tsükkel ning erinevate andmestruktuuride manipuleerimine. Eksamil on tavaliselt sellised ülesanded andnud 3 punkti.
- Keskmised ülesanded on sellised, mida keskmine tudeng lahendab kuni 30 minutit ning sõltuvalt ülesande tüübist võib koodiridade arv olla kuni ~25 raksematel juhtudel. Näiteks mängu kivi-paber-käärid loogika programmeerimine. Need ülesanded eeldavad andmestruktuuride valdamist, lihtsama objektorienteeritud koodi ja rekursiooni kirjutamise oskusi. Sellised ülesanded on tavaliselt andnud 5 punkti.
- Rasked ülesanded eeldavad tudengilt head oskust objektorienteeritud koodi kirjutada ning selliste ülesannete piires ka keerulisema loogika realiseerimist. Näiteks ülikooli õppeinfosüsteemi lihtsustatud variandi loomine, kus osapoolteks on tudeng, ülikool ning aine. Mõned rasked ülesanded eeldavad algoritmide teadmisi. Sellised ülesanded on tavaliselt andnud 10 punkti eksamil.

Iga kategooria all on nimekiri ülesannetest. Iga ülesande kausta sees asub tekstifail `.rst` formaadis, mille sees on ülesande seletustekst koodinäidete, inglisekeelse Javadociga ning oodatava funktsiooni nime ja parameetritega. Kirjeldusfaili illustreerib joonis number 2. Teiseks failiks on testifail vastavalt programmeerimise keelele, milles sisalduvad kirjeldusfailis oleva ülesande testid. Testiklassis on Java puhul sisse toodud välise teenusena kasutatav StudentTester. Testid on kirjutatud testNG testimisraamistiku abil. Iga testifail on sarnase struktuuriga, kuna sisaldab meetodeid `testExample()`, mis enamasti

[18] Loodud Gitlab repositoorium [WWW] https://gitlab.cs.ttu.ee/Deniss.Potapenko/coding_challenges

kontrollivad `.rst` failis olevaid näidissisendeid ning `testRandom()`, mis kontrollib juhuslikke sisendeid ja väljundeid. Muud meetodid katavad erinevaid ülesande keskseid spetsiifilisemaid stsenaariume.

Juurkaustas on lisaks raskuskategooriatele *README.md*, mis selgitab ülesannete kategooriateks jagamise põhimõtteid. Samuti on selles failis kirjeldatud ülesannete lisamise juhend. See sisaldab kirjeldust, kuidas `.rst` faile hallata ning õpetust, milliseid printsiipe tuleks jälgida testide koostamisel juhul, kui soovitakse uusi ülesandeid kogusse lisada.

Lisaks on juurkaustas olemas kaust „Lisad", milles võib leida aastate kaupa välja toodud ülesannete lahendusi. Veel on „Lisad" kaustas tekstifail, kuhu on kirja pandud ideed selle kohta, milliseid ülesandeid oleks võimalik valmis teha ja kogusse lisada. Samas kaustas asuv fail „Lisatud testid ja parandused.rst" kirjeldab kõiki muudatusi, mida autor ülesannete koostamisel sisse viinud. Viimasena on „Lisad" kaustas fail „Lühikirjeldused ja statistika.rst", mille sisu on kirjeldatud järgmises peatükis.

2.4. Korrastatud testide statistika

Töö käigus on loodudtabel, mida saab lugeda antud dokumendi „Lisa 1" all. „Lühikirjeldused ja statistika.rst" sisaldab endas üldistavat informatsiooni iga ülesande kohta, mis kogus asub. Tabeli veerud kirjeldavad järgmisi ülesande omadusi:

- Ülesanne – vastab ülesande unikaalsele nimele, mis peab piisavalt täpselt kajastama ülesande sisu.
- Kategooria – põhilised ülesannet iseloomustavad võtmesõnad. Tavaliselt on kirjeldatud andmestruktuurid, mida ülesandes kasutatakse (sõne, massiiv, järjend, kujutis). Veel on iseloomustamiseks kasutatud selliseid võtmesõnu nagu rekursioon, OOP, juhul kui tegemist on objektorienteeritud ülesandega, matemaatika, juhul, kui tegemist on matemaatilise kontekstiga ülesandega, algoritm, kui ülesanne eeldab algoritmide kirjutamist. Lisaks on muid spetsiifilisemaid võtmesõnu eri tüüpi ülesannete kirjeldamisel.

- Keskmine % – keskmine protsentuaalne tulemus, mis on saadud ülesande eest kõikide tudengite poolt, kes on seda ülesannet lahendanud. Andmed pärinevad 2015-2017 aastate Moodle hindamisraamatust.
- Inimest eksamil/0p saanud – Kuna antud töö käigus on töödeldud „Java programmeerimise põhikursuse“ eksamiülesandeid, siis antud tabeli veerg määrab ära mitu inimest on osalenud eksamil, kus see ülesanne sees oli, ning mitu inimest said vastaval eksamil selle ülesande eest 0 punkti või ei teinud seda ülesannet üldse. See veerg aitab määrata keskmise protsendi veeru objektiivsust, kuna valim on erinevate ülesannete puhul erinev.
- Pärineb – klassifikaator, mis määrab ära, millisest kogust algselt ülesanne pärineb. Eesliide „E“ tähendab, et tegemist on eksamiülesandega. Selle järgneb aasta, mil seda ülesannet kasutati. Eesliide „EX“ tähendab, et tegemist on semestriülesandega.
- Kasutatud platvormil – Kuna osa ülesandeid agregeeritud kogust on kasutusel loodud harjutamisplatvormil, siis on selles veerus määratud ülesanded, mis on platvormi kaudu avalikuks tehtud. Muud ülesanded ei ole avalikud ning neid saab vajadusel eksamitel kasutada.

Lisaks tabelitele on selles failis numbriline ülevaade kõikidest ülesannetest, mis on agregeeritud kogus olemas. 21.05.2018 seisuga on kogus 78 ülesannet (tekstid + testid) ja 15 ülesannet 2015 aasta eksamist (tekstid).

3. KOODI KIRJUTAMISE HARJUTAMISPLATVORMI ARENDAMINE

Järgnevalt on esitatud koodi kirjutamise platvormi analüüs, milles on kirjeldatud rakenduse nõuded, töö spetsifikatsioon ning rakenduse kasutusjuhud. Seejärel on kirjeldatud protsessi, kuidas nõuete realiseerimiseni jõuti ning metoodikaid, mida kasutati, et töö võimalikult ladusaks teha. Lõpuks kirjeldatakse platvormi loomisel saavutatud tulemus, ehk vastavus esitatud nõuetele. Lahti on selgitatud veebirakenduse funktsionaalsus ning lähteprojekti struktuur.

3.1. Rakenduse analüüs

Selles peatükis kirjeldatakse veebirakenduse funktsionaalseid ja mittefunktsionaalseid nõudeid selgitatakse töö spetsifikatsiooni. Lõpuks tuuakse välja mõningad rakenduse kasutusjuhud.

3.1.1. Rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded

Funktsionaalseid nõudeid vaadeldakse koodi kirjutamise platvormi kasutaja seisukohalt.

Kasutaja saab teostada järgmisi toiminguid:

- Pääseda platvormile ligi veebibrauseri abil
- Valida keel, milles programmeerimiseülesandeid lahendada soovitakse
- Valida ülesande lahendamise kategooria vastavalt valitud keelele
- Valida kindel ülesanne lahendamiseks suuremast ülesannete nimekirjast ning nimekirjas näha iga ülesande raskusastet
- Näha valitud programmeerimisülesande kirjeldust eesti ning inglise keeles, sisend-väljundeid ning malli õige funktsiooni nimega.
- Kirjutada lähtekoodi selleks otstarbeks loodud koodiaknasse ning saada koodi kirjutamisel redaktorilt mõningat süntaktilist abi
- Esitada kood ning saada automatiseeritud tagasisidet koodi õigsuse kohta

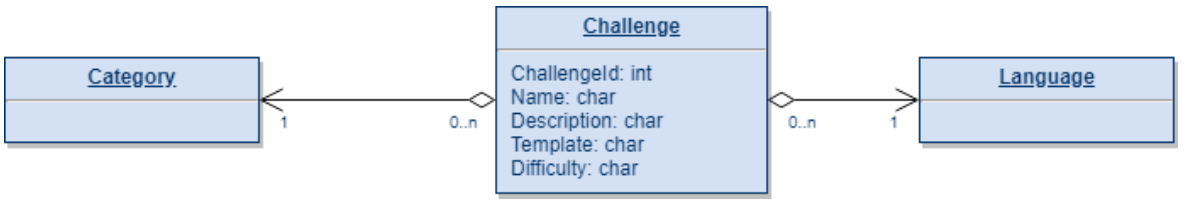
Mittefunktsionaalsete nõuete alla käivad inglisekeelne kasutajaliides ning lähtekood. Lõputöö kirjutamise ajal töötab rakendus viimase versiooni Firefox, Chrome, Safari ja Microsoft Edge brauserites.

3.1.2. Töö spetsifikatsioon

Loodav harjutamisplatvorm võimaldab kuvada kindlal kujul olevaid programmeerimiseülesandeid veebilehel. Iga ülesanne koosneb pealkirjast, kirjeldusest ja mallist. Malle kuvatakse tekstilahtris, kuhu on võimalik oma ülesande lahendus sisse kirjutada. Seejärel saadetakse lahendus ning viide ülesandele välise teenusena kasutatud testimissüsteemi, mis tagastab vastuse ehk testide tulemused otse veebirakendusele, mida saab lahendajale koheselt kuvada.

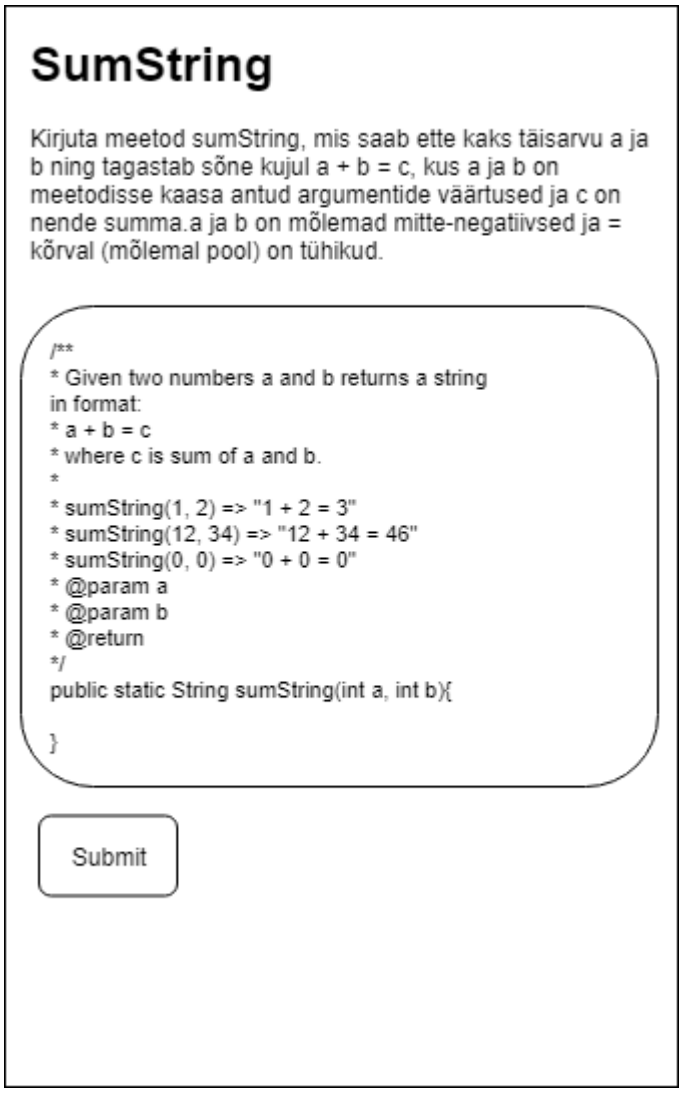
TTÜs kasutatakse programmeerimisainetes automaatteste selleks, et tudengite programmi tööd revideerida ning kontrollida nõuetele vastamist. Selle tarvis on TTÜ jaoks loodud testimissüsteem, millest uuendatud versiooni nimega Arete't kasutatakse antud töös välise teenusena.

Loodavas veebirakenduses kuvatakse ülesannete nimekiri, mida tudeng saab lahendada. Ülesannete struktuuri illustreerib diagramm number 3.



Joonis 3 Ülesande isendi olemi-suhte diagramm

Ülesannete hoidmiseks kasutatakse põhitabelit `challenges`, mis viitab ülesandele. Iga ülesanne on iseloomustatud unikaalsete numbrilise identifikaatoriga ning ülesande nimega, mis peab võimalikult täpselt kajastama sisu. Näiteks ülesanne `SumString`, tähistab, et tegemist on sõne summeerimisega. Ülesande kirjeldus on eestikeelne juhend, mis seletab täpset oodatavat programmi käitumist. Ülesande malliks on inglisekeelne Javadoc või Pythondoc, mis sisaldab endas tõlgitud inglisekeelset ülesande kirjeldust ning sisendi ja vastava väljundi paare. Samuti on mallis kirjas ka funktsiooni nimi ja vastav parameeter, mis funktsiooni muutujana oodatakse. Java puhul kassi definitsiooni ei nõuta. Veebirakenduses kuvatavat ülesannete ülesehitust illustreerib joonis number 4.



Joonis 4 Konkreetse ülesande struktuur veebirakenduses

Kuna suur osa platvormile lisatavetest ülesannetest on võetud korrastatud ning agregeeritud 2015-2017 aastate eksamiülesannete kogust ning seal on jagatud ülesandeid raksusastme järgi, siis harjutamisplatvormi arendamisel on seda lähenemist jätkatud. Raskusastmed jagunevad kolmeks: lihtne, keskmine ja rakse. Lihtsad ülesanded nõuavad oskusi vähesel määral ning nõuavad maksimum kuni 10 rida koodi. Keskmised ülesanded nõuavad paremat keele valdamist. Rasked ülesanded on tüüpiliselt palju pikemad ning nõuvad head andmestruktuuride ja algoritmide teadmist. Selle tarvis on ülesandele lisatud väli `Difficulty`.

Lisaks eelnevalt mainitud väljadele on `challenges` tabelis kaks välisvõtit, mis viitavad tabelite `languages` ja `categories` primaarvõtmetele vastavates klassifikaatoritest tabelites. Kuna loodav platvorm peab võimaldama lahendada ülesandeid erinevates keeltes ning TTÜs enamasti ei kasutata ülesannete puhul puhast sisendi ja väljundi kontrollimist,

siis peavad ülesanded olema programmeerimiskeele järgi eristatavad. See võimaldab vajadusel saata päringuid erinevatesse testritesse selle järgi, milline on ülesande keel. Algselt on realiseeritud kahe keele toetamine: Python ja Java.

Kategooriate järgi eristamine on oluline, kuna võimaldab tudengitel lahendada neid ülesannete kategooriaid, mis teda huvitavad. Kategooriad on erinevates programmeerimiskeeltes erinevad, kuna näiteks keeles Python ei ole andmestruktuuri nimega `map`, vaid selle asemel andmestruktuur nimega `dictionary`.

Lisaks ülesannete kuvamise funktsionaalsusele võimaldab veebirakendus ülesande lahendust saata testimissüsteemi. Testimissüsteem on kasutusel valmisproduktina ning harjutamisplatvormi autor ei ole testrit loonud. Testimissüsteemi tulemuse saatmine tähendab korrektse JSON sõne saatmine päringuna testri *endpointi*.

Arvestades testri poolt seatud reeglitega koosneb testrisse lahenduse saatmine kahest etapist: testide üleslaadimine, ehk testide asukoha määramine ning lahenduse saatmine testimiseks. Mõlemad etapid on järgnevalt lahti kirjutatud.

TTÜ testimissüsteem Arete töötab põhimõttel, et iga ülesande testifail asub kindlas Git salves ning kindlas kaustas. Seepärast peab iga ülesande lisamisel veebirakendusse tegema ühekordse `POST` päringu testimissüteemi *endpoint*'i, mis viitaks salvele, kust testifail pärineb. Vastav JSON päring on näidatud joonisel 5.

```
{
    "name": "SumString",
    "course": "154858iapb_test",
    "tester": "javang",
    "testerExtra": "stylecheck",
    "unitTests": {
        "repositoryUrl":
"git@gitlab.cs.ttu.ee:Deniss.Potapenko/154858iapb_test.git"
    }
}
```

Joonis 5 Testide üleslaadimine testrisse konkreetse näite baasil

Testifailide hoidmise tarvis on loodud Gitlab salv [19]. Päringu väli `name` on unikaalne identifikaator ja iseloomustab seose veebirakenduse ülesande nime ning viidatava testi nime

[19] Testide hoidmise Gitlab salv [WWW] https://gitlab.cs.ttu.ee/Deniss.Potapenko/154858iapb_test

vahel. `course` on Git projekti nimi, kust hakatakse teste otsima. Väljas `tester` määratakse ära, mis keele testrisse vastus saadetakse. `repositoryUrl` määrab ära juursalve, kust hakatakse projekti otsima. Testimissüsteemil on SSH kasutaja, mis on testifailide Gitlab repositooriumisse lisatud. Selle abil saab Arete testifailide salve turvaliselt alla kloonida.

Kui kõik olemasolevad ülesanded on testri jaoks ära kaardistatud, siis saab saata päringuid lahendustega, mida illustreerib joonis number 6.

```
    "source": {
        "files": [
            {
                "path": "Challenge.java",
                "content": "Class Challenge{ public static String
sumString(int a, int b){return "test";}}"
            }
        ]
    },
    "project": {
        "name": "SumString",
        "course": "154858iapb_test"
    }
}
```

Joonis 6 Ülesande lahenduse saatmise päring Java näite baasil

Olulisteks väljadeks on `path`, mis määrab ära klassi, kus lahendus asub. `Content` väli kannab endaga kaasas lähtekoodi, mida tudeng kirjutas. Kogu kood on pandud klassi, mis peab olema sama nimega, nagu `path` väljas olev klassi nimi. Klassi sisse pakendamine on iseloomulik Java, aga mitte Python projektidele. `Project` väljad määravad ära millisest Git salvest vastava nimega ülesannet otsida.

Lahendusega päringule saadb testimissüsteem sünkroonse vastuse JSON kujul, milles on palju metainfot käivitatud testide kohta, kompilatsiooniprotsessi kohta ning algse päringu kohta. Selle töö raames peale töötlemist kasutatav JSON osa on välja toodud joonisel 8.

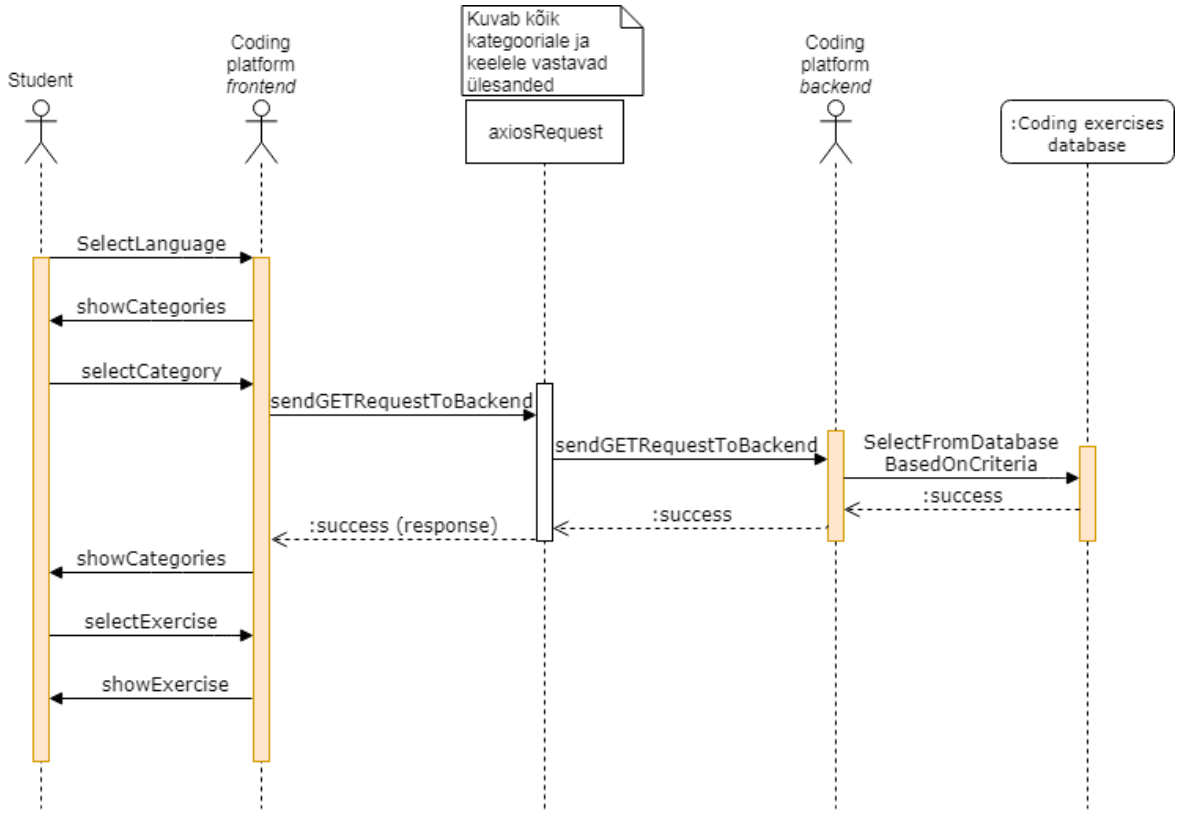
```
"output": "TEST RESULTS\n\n\n\nCompilation succeeded.\n\n\n ---\nTestSumString
(TestNG)\nThu May 17 11:03:03 UTC 2018\n ---\n\nPassed unit tests: 3/3\nFailed
unit tests: 0\nSkipped unit tests: 0\nGrade: 100.0%\n\nOverall grade:
100.0%\n"
```

Joonis 7 Testrisüsteemilt saadud vastuse tulemust määrav JSON väli

3.1.3. Rakenduse kasutusjuhud

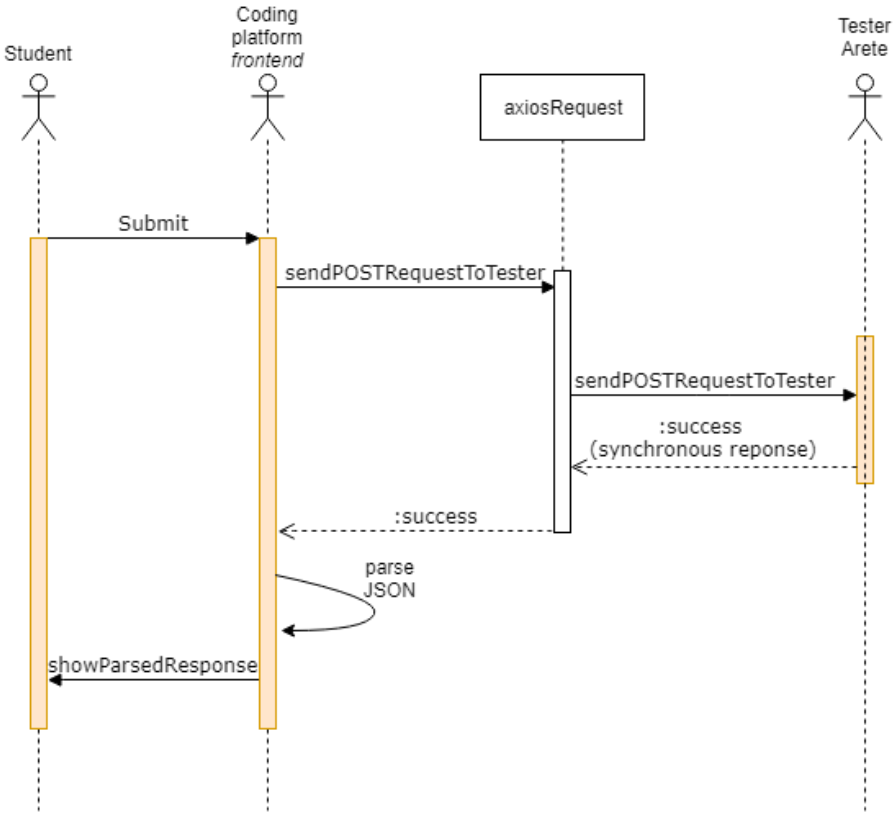
Järgnevalt tuuakse näiteid mõnest kasutusjuhust ning nende kohta käivatest jadadiagrammidest. Näited on tudengi poolt valitud ülesandeni jõudmine ning lahenduse esitamise protsessi kohta.

Joonis 8 kujutab valitud ülesandeni jõudmise protsessi. Veebirakendus kuvab tudengitele sisseehitatud valiku keeltest. Keele valiku järgselt kuvatakse tudengile vastavale keelele sobivad kategooriad. Tudeng valib kategooria ning selle pealt saadetakse `HTTP GET` päring serveripoolele, mis omakorda saadab päringu otse andmebaasi. Seejärel saadetakse serveri poolt vastus tagasi veebirakendusse, mis omakorda kuvab ülesanded vastavalt valitud keelele ja kategooriale. Vajutades valitud ülesandele ei saadeta enam andmebaasi päring vaid kuvatakse detailvaade ülesandest, mille unikaalne nimi vastab andmebaasist sisse loetud nimistu kindla ülesandega.



Joonis 8 Ülesande valimise protsess jadadiagrammina

Joonisel 9 jätkatakse ajaliselt joonisel 8 olevat tegevust ning saadetakse ülesande lahendusega HTTP POST päring selliste JSON väljadega nagu on kujutatud joonisel 6. Kui testrisüsteem saab päringu kätte siis saadab ta sünkroonselt vastuse tagasi veebirakendusse. Veebirakendus töötleb JSON vastuse ning ilmutab tudengile testi tulemused.



Joonis 9 Testrisse lahenduse saatmise jadadiagramm

3.2. Harjutamisplatvormi arendusprotsess ning kasutatud metoodikad

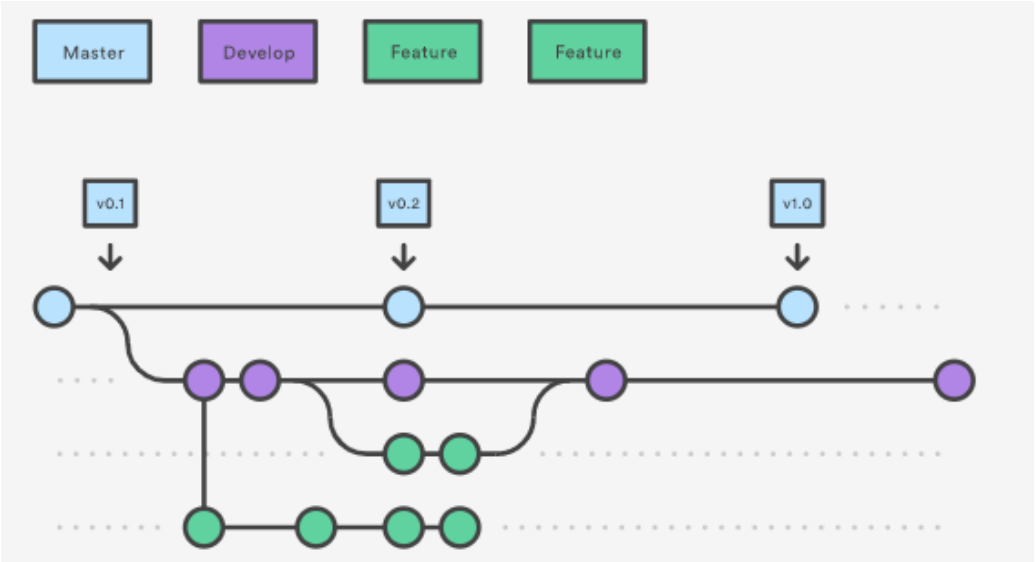
Platvormi loomisel on eesmärgiks järgida laialt levinud tarkvaraarenduse metoodikaid ning mustreid. Evitusprotsess peab olema võimalikult automatiseeritud kasutades sellist tehnoloogiat nagu Jenkins. Serveris on rakendus lihtsasti käivituv ning masinast isoleeritud Docker konteineri abil. Versioonihaldusena on kasutatud Giti, millele on rakendatud arendusmuster, mis hoiab lahus arenduskeskkonda ning *live*–keskkonda.

3.2.1. Gitflow arendusmuster versioonihalduses

Gitflow puhul on tegemist Vincent Driesseni poolt väljapakutud arendusmustriga, mis kasutab versioonihaldussüteemi Git. [9] Tegemist on abstraktse õpetusega, kuidas võiks oma Giti haldust mõistlikult korraldada. Gitflow kirjeldab semantiliselt ära, kuidas ning millal harud (*branch)* peavad omavahel suhtlema, kuidas neid tekitada ning hallata.

Lisaks peaharule (*master*) kasutab Gitflow ka arendusharu (*develop)*. Peaharu ülesandeks on hoida endas lõpptoodet. Arendusharu eesmärgiks on hoida kogu projekti muudatuse ajalugu. Uute funktsionaalsuste arendamisel luuakse arendusharust uus funktsionaalsuse haru

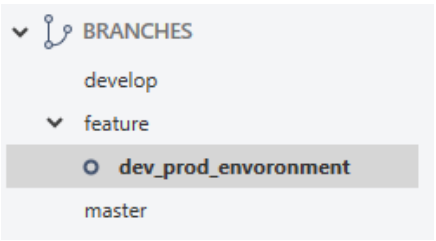
(*feature*). Kui funktsionaalsuse harus on kogu töö tehtud, siis mestitakse (*merge*) see arendusharru.



Joonis 10 Gitflow arendusmustri graafiline esitus [20]

Kui on saabunud hetk, kui töötavad funktsionaalsusi on piisavalt palju, et valmistootest uus versioon välja lasta, siis luuakse uus haru, mille nimeks on *release*, ehk väljalaskeharu. Kui väljalaskeharus olev programm on läbitestitud, siis mestitakse see põhiharusse ning jätkatakse tööd arendusharus. Graafiliselt on arendusmuster näidatud joonisel 10.

Antud töös kasutatakse põhiharu ja arendusharu. Põhiharu on serveris töötav rakendus ning arendusharus käib arendamine. Samuti kasutatakse suuremate tükkide puhul funktsionaalsuse harusid nagu on näha joonisel 11.



Joonis 11 Arendusharude struktuuri näide antud töös

3.2.2. Dockeri kasutamine rakenduse konteineriseerimiseks

Sel hetkel, kui töötav JAR fail rakendusega on kokku ehitatud, siis on võimalik minna servermasinasse, et seal teha Docker pilt ning sellest panna püsti konteiner, kus rakendus

[20] https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

töötama ning päringutele vastama hakkab. Dockeri kasutamise põhjus ning eelised on kirjeldatud peatükis 1.3.5.

Selleks, et konteiner püsti panna ning rakendusserver käivitada, piisab JAR failist ning `Dockerfile`'st servermasina samas kaustas. `Dockerfile` puhul on tegemist konfiguratsioonifailiga, kus antakse ette alusprojekt `FROM` klauslis – antud juhul JDK. Samuti kirjeldatakse Dockefile's projekti JAR fail, argumendid, millega see JAR fail käivitatakse ning konteineri sisse kantavad või tekitatavad kasutad. Antud töös käivituv `Dockerfile`'i sisu on välja toodud joonisel number 12.

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ADD target/backend-0.0.1-SNAPSHOT.jar app.jar
ENV JAVA_OPTS=""
ENTRYPOINT ["java","-
Djava.security.egd=file:/dev/./urandom","-
Dspring.profiles.active=prod","-jar","/app.jar" ]
```

Joonis 12 `Dockerfile` sisu antud töös

Pärast projekti koostet saab projekt käima panna andes käivituskäsule ette argumente. Antud töös on käivituskäsk järgmine:

```
docker run -v /home/deniss/codingbat /config/:/config -p -d
80:8080 codingbat
```

Olulisemateks on pordi 8080, milleks on rakenduse port kaardistamine pordile 80, mis on HTTP port ning Spring konfiguratsioonifaili konteinerisse lisamine *config* kasuta alla. Konfiguratsioonifail sisaldab tundlikku infot andmebaasi paroolide kohta ning seda ei maksa salves hoida. Teiseks saab läbi välise konfiguratsioonifaili palju mugavamalt serveripoolt muuta ilma uue JAR faili loomata.

3.2.3. Jenkinsi abil evitusprotsessi automatiseerimine

Kuigi peatükis 3.2.2 kirjutatu on õige viis Docker konteineri manuaalsel käivitamisel, siis on palju mugavam teha evitusprotsess automaatseks nii, et Gitlab repositooriumisse uue koodi ülespanemisel ning Jenkins koosteprotsessi alustamisel käivituks uus konteiner automaatselt.

Erinevates etappides täidab `Jenkinsfile` erinevaid ülesanded. Alguses kloonib ta alla kindlalt määratud repositooriumi ning käivitab selles repositooriumis `mvn clean`

`install` käsu, mis tõmbab alla vajalikud `NPM` moodulid, kannab nad üle serveripoole ressursside alla ning tekitab sellest JAR faili. Kuna `Dockerfile` ja tekkinud JAR fail asuvad `backend/target` kaustas, siis `Dockerfile:build`, käsk tekitab pildi (*image*), mis on taaskasutatav hetktõmmis konteinerist, mida käivitatakse *backend* kataloogis. Seejärel `thesis/codingbat` nimeline pilt salvestatakse arhiivis ning laaditakse uuesti sisse rakendusmasinas, kus eelnevalt kustutatakse sama nimega pildid ja konteinerid. Seejärel käivitatakse sama `run` käsk, mis oli välja toodud peatükis 3.2.2. Nüüd jääb käivitatud konteiner tööle rakendusmasinas. Antud töös on evitusprotsessi kirjeldav `Jenkinsfile` joonisel 13.

```
node('master') {
  try {
    stage "Checkout repositories"
      // Clone git repository form Gitlabs
     checkout scm

    stage "Build docker container"
      // Install frontend/backend and convert JAR to image
     sh(returnStdout: false, script: 'mvn clean install -DskipTests')
     sh(returnStdout: false, script: 'mvn  --projects backend
dockerfile:build')

    stage "Deployment"
      sh(returnStdout: false, script: 'docker save -o codingbat.tar
thesis/codingbat')
      withCredentials([[$class: 'FileBinding', credentialsId: '***',
variable: 'SSHKEYFILE']]) {
      autorunCommand = 'docker run -v /home/deniss/thesis/config/:/config -
d -p 80:8080 --name codingbat thesis/codingbat'
        sh 'eval `ssh-agent -s`; ssh-agent bash -c \'ssh-add "' +
env.SSHKEYFILE + '";scp codingbat.tar jenkins@IP: && ssh jenkins@IP "docker
load -i codingbat.tar; docker kill codingbat; docker rm codingbat;docker
images | grep -oP \\"(?<=none)[ ]*[0-9a-z]*\\" | grep -oP \\"[0-9a-z]*$\\"
| xargs -I hash docker rmi hash;' + autorunCommand + ';echo \\"Deployment
finished!\\""\''
      }
      // docker save -> tar-file
      // scp tar-file into server
      // docker run
    stage "Cleanup"
      // remove old images etc.
  } catch(err) {
    currentBuild.result = 'FAILURE'
    throw err
  } finally {
      // step([$class: 'Mailer', notifyEveryUnstableBuild: true,
recipients: 'g@gmail.com', sendToIndividuals: true])
  }
}
```

Joonis 13 Töös kasutatav `Jenkinsfile`

3.3. Rakenduse funktsionaalsus

Töö tulemusena on loodud programmeerimise harjutamisplatvorm, kus on võimalik ülesandeid veebirakenduses lahendada ning koheselt väliteenusena kasutatavalt testimissüsteemilt tagasisidet saada. Joonised kasutajaliidesest on „Lisad 2“ all. Kasutajaliides kasutab *Material Design* [21] komponenete, mille on teinud kättesaadavaks Vue js komponentraamistik Vuetify js.

Tudeng saab valida soovitud ülesannete lahendamise keele. Seejärel kuvatakse talle sellele keelele vastavad kategoorad. Kategooriad on ülesannete tüüpide klassifikaatoriteks. Tavaliselt määravad nad ära andmestruktuure, mida ülesandes kasutatakse või muid iseloomulike jooni. Kategooriad on võetud otse antud lõputöö esimeses osas valmis agregeeritud kogu ülesannete kategooriatest.

Kategooria valiku järel järgneb vastava keele ja kategooria ülesannete valik, mis on salvestatud andmebaasi ning mis pärinevad agregeeritud kogust. Iga ülesande juures on määratud ära ülesande raskusaste, mis on jaotatud kolme võimalikku väärusesse: lihtne, keskmine ja raske. Uusi ülesandeid saab lisada andmebaasi `POST` päringu abil, mis saadab päringu tagakomponenti ning too omakorda andmebaasi või manuaalse `INSERT` lause kirjutamisel.

Iga ülesanne on varustatud ülesande nimega, kirjeldusega, näite sisend-väljund paaridega ning inglisekeelse Javadociga. Konkreetse ülesande lahendamisele aitab kaasa brauseripõhine tekstiredaktor Codemirror [22] , mis on spetsialiseeritud lähtekoodi kirjutamisele. Rakendusega tulevad kaasa funktsionaalsus, mis aitab koodi kirjutamist mugavamaks teha. Antud töös aitab Codemirror süntaksit esile tõsta ning koodiridu, taandeid ja sulgusid määrata.

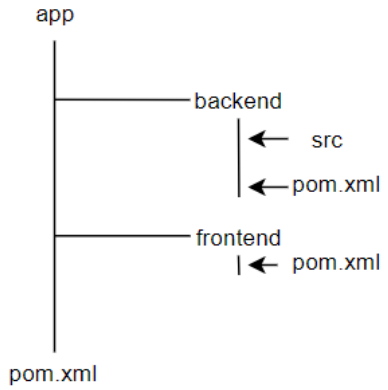
Valminud kood saadetakse TTÜ Arete nimelisse testimissüsteemi, kus on ära määratud koodiplatvormiga seotud ülesanded. Vastus tuleb sünkroonselt, ehk tudeng peab vastuse ära ootama. Keskmiselt läheb ühe vastuse saamiseks *circa* 7 sekundit. Platvorm töötleb testrist saadud JSON tulemust ning see kuvatakse kasutajaliideses. Testri vastus on ekvivalente programmeerimise ainetes elektronpostile saadud vastusega ning see kajastab tulemusprotsenti ja õnnestunud ning ebaõnnestunud teste koos sisuga.

[21] Material desgin standard [WWW] https://material.io/guidelines/
[22] Codemirror veebileht [WWW] http://codemirror.net/

3.4. Projekti struktuur ning kooste

Tagamaks madala sõltuvuse ning kõrge kokkukuuluvuse on loodud kaks erineva vastutusalaga moodulit. *Backend* moodul sisaldab endas kogu Spring Boot raamistiku abil loodud lähtefailide klassistruktuuri. Projekti *frontend* moodul sisaldab endas Vue.js raamistiku lähtekoodi, mis on omakorda jaotatud komponentideks ning aitab hoida erinevaid rakenduse osasid võimalikult lahus.

Lõpuks kokku ehitatav projekti arhiiv peaks sisaldama tervet käivitatavat projekti. Moodulite haldamiseks ja kooste korrektsust määrab Maven, mis kasutajaliidesest (*frontend)* ning serveripoolest (*backend*) paneb kokku ühise käivitatava JAR faili. Struktuuri illustreerib joonis 14.



Joonis 14 Projekti Maven struktuur

Maven projekti üldnimeks on *app*, mille sees olev pom.xml konfiguratsioonifail määrab ära, kuidas projekt kokku ehitada. Selle sees asub üldine `pom.xml`, mis käsib projekti ehitada kahest väiksemast moodulist, mille nimedeks ongi *backend* ja *frontend.*

Kasutajaliidese mooduli puhul peab ta kooste käigus alla laadima lokaalselt Node.js ja temast sõltuvuses oleva NPM pakettide haldaja, selleks et servermasinas saaks Vue.js rakendust käivitada. Selleks, et hoida võimalikult lahus eeskomponendi ja tagakomponendi mooduleid on Maveni jaoks loodud eriline laiendus frontend-maven-plugin [23], mis võimaldab Maven kooste ajal Node.js ja tema mooduleid kasutada ilma globaalselt neid installeerimata, vaid kasutades lokaalset projektipõhist installatsiooni. Kogu *frontend* `pom.xml` fail sisaldab endas frontend-maven-plugin'i poolt võimaldatud käivituskäske,

[23] Frontend maven plugin Github [WWW] https://github.com/eirslett/frontend-maven-plugin

nagu `npm install` ja `npm run build`, mis tekitab kokku pakitud rakenduse versiooni *dist* kataloogi *frontend* moodulis.

Oluline on ka ära märkida, et enne `npm install` käsu käivitamist kasutatakse maven-clean-plugin'it selleks, et rakenduse serveris vältida ülekirjutamisega seotud konflikte. Antud töös ilma selleta rakendusserver tekitas *Access Denied* vea.

Backend moodulis kopeeritakse `maven install` käigus *frontend'*i poolt loodud `dist` kataloogi sisu ja pannakse ta *backend*'is `resources` kataloogi alla, kuna ainult selles kataloogis olevaid ressursse Spring Boot näeb ning oskab kasutada.

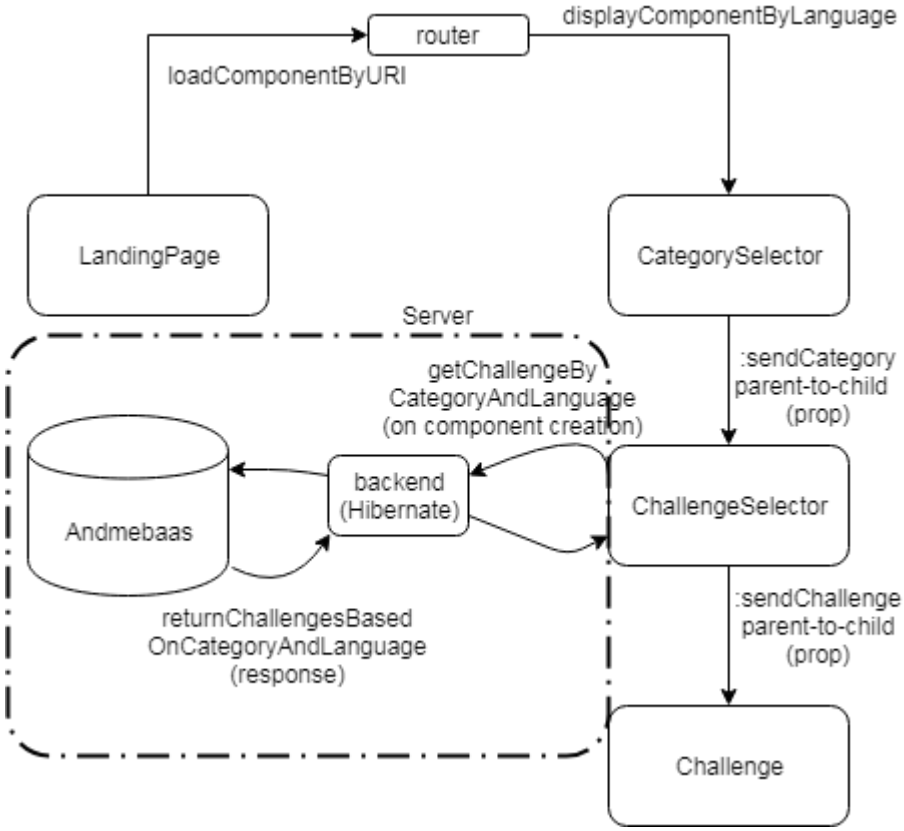
Teiseks oluliseks osaks serveripoolses `pom.xml` failis on integratsioon Dockeriga. `Dockerfile-maven-plugin` [24] võimaldab `mvn clean install` ajal tekitada traditsioonilise JAR faili asemel töötava Docker pildi. Selle tarvis on vaja projekti juurkaustas teha eelnevalt mainitud pistikprogrammi jaoks kättesaadavaks soovitud `Dockerfile` dokument. See funktsionaalsus on vajalik automaatse evitusprotsessi jaoks Jenkins'i abil.

Backend mooduli eesmärk on ülesannete sisse lugemine andmebaasist vastavate *frontend* päringute tagajärjel. Moodul sisaldab endas kolme põhilist paketti. `Repository` sees olev liides võimaldab andmebaasist pärida soovitud kriteeriumile vastavad read. Selle töö raames on vajalik teha päringuid, mis tagastavad ülesandeid vastavalt kategooriale ning valitud keelele. `Entities` pakett sisaldab endas klasse, mis on kaardistatud Hibernate abil vastavateks andmebaasi tabeliteks. `Controller` pakett sisaldab *endpoint*'ide nimekirja, mida rakendus veebi paljastab. Selle abil on võimalik uusi ülesandeid andmebaasi lisada ning ülesandeid vastavalt kriteeriumidele pärida.

Vue js poole peal on oluline `App.vue`, mis annab veebilehele struktuuri ning kuvab komponente vastavalt vue-router'i poolt määratud sätetele. Komponentide ja URI teekondade kaardistus on määratud `router/index.js` failis. `networking` kaust on axiose [25], ehk HTTP kliendi sätete kataloogiks, kus on määratud põhilised URL'id, kuhu päringuid saadetakse.

[24] Dockerfile maven plugin github [WWW] https://github.com/spotify/dockerfile-maven
[25] Axios HTTP kliendi Github [WWW] https://github.com/axios/axios

Komponentideks jagamine on teostatud `components` kataloogis, kus on hulk erinevate veebirakenduse osade eest vastutavaid komponente. Mõned komponendid ei muutu kogu rakenduse elutsükli ajal ning on kogu aeg olemas. Sellisteks komponentideks on `NavigationBar.vue` ja `Footer.vue`. Muud komponendid vahetuvad sõltuvalt kasutaja valikutest nagu on illustreeritud joonisel 15.



Joonis 15 Projekti sisene andmevahetus ja struktuur

`LandingPage.vue'il` on valik keeltest, mida vajutades saadetakse kasutaja `/'keel'` *endpoint*'i. Näiteks, valides keele Java on URI lõpp `/java`. Seejärel kuvatakse komponent `CategorySelector.vue`, mis saab lisainfona keele kätte. Edasi sõltuvalt vajutatud kategooriale saadetakse info vajutatud kategooria nime kohta edasi `ChallengeSelector.vue` komponenti. Nüüd on sellel komponendil olemas info keele ja valitud kategooria kohta. Vastav `GET` päring saadetakse andmebaasi, kust tuleb vastusena JSON järjend ülesannetest, mis vastavad valitud keelele ja kategooriale. Kui kogu info on käes, saadetakse alamkomponenti `Challenge.vue` kogu info klikitud ülesande kohta. Kogu järjendi sisselaadimise eelis võrreldes ühe ülesande laadimisega seisneb selles, et uue ülesande valimisel samast keelest ja kategooriast ei pea andmebaasi päringut tegema.

KOKKUVÕTE

Töö esimeseks eesmärgiks oli luua agregeeritud ning kategoriseeritud ülesannete kogu, kuhu kuuluvad varasemalt kasutatud programmeerimisülesanded. Teiseks eesmärgiks oli luua TTÜ tarvis koodikirjutamise platvorm, kus on võimalik välja pandud ülesandeid lahendada ning koheselt automaattestide abil tagasisidet saada.

Esimese eesmärgi saavutamiseks loodi ühtne Gitlab salv, kuhu koondati Java eksamiülesanded aastatest 2015-2017. Agregeerimise käigus jagati ülesanded kolme raksusastmesse, struktureeriti ülesannete teste ning lisati juhuteste. Samuti migreeriti kõik vanemad ülesanded testNG testimisraamistikku ning integreeriti kõik ülesanded StudentTesteriga. Kõikidele ülesannetele koostati reStructuredText formaadis kirjeldus ning näidiskood eesti ja inglise keeles. Samuti koguti kokku kolme aasta ülesannete lahendused ühte kausta ning tekitati lühikirjelduse ja statistika leht, kus on võimalik näha ülesannete kohta käivat informatsiooni ning kogutud statistikat.

Töö käigus on korrastatud 93 Java eksamiülesannet, millest 78 ülesannet on korrastatud teksti ja testidega ning 15 ülesannet ainult tekstiga.

Teise eesmärgi saavutamiseks loodi Spring Boot ning Vue js moodulitest koosnev harjutamisplatvorm, kus ülesanded on jagatud programmeerimiskeeltesse (Java ning Python) ja kategooriatesse. Rakenduses kasutatakse ülesandeid, mis lisati ülesannete kogusse. Ülesannete tekstid loetakse sisse loodud MySQL andmebaasist ning Codemirror tekstiredaktori koodiaknaasse kirjutatud kood saadetakse TTÜ uute Arete-nimelisse testimissüsteemi, kust testimise tulemus saadakse. Vastus kuvatakse kasutajale sünkroonselt pärast testimisesüstemist vastuse tulekut. Visuaalse osa eest vastutab Vue.js komponentraamistik Vuetify.js, mis pakub kujunduskomponente. Töö käigus seadistatakse automaatne evitus ja konteineriseerimine. Selleks kasutatakse Jenkinsit ja Dockerit.

Agregeeritud ülesannete seast kasutati ükskuid 2018. aasta põhikursuse ülesannetena. Ülesandeid plaanitakse kasutada ka sama aine eksami korraldamisel. Harjutamisplatvormi on võimalik edasi arendada lisades sisselogimise ja laiendades ülesannete ning keelte valikut.

Kogu korrastatud ülesannete lähtekood on laetud TTÜ Giti salve aadressil https://gitlab.cs.ttu.ee/Deniss.Potapenko/coding_challenges ning harjutamisplatvorm aadressil https://gitlab.cs.ttu.ee/Deniss.Potapenko/thesis

KIRJANDUSE LOETELU

[1] Lauren Csorny „Careers in the growing field of information technology services“ *Beyond the numbers* vol. 2 / no. 9 April 2013. (12.04.2018)

[2] Adam Heitzman. 2018. 8 Innovative Industries That Will Really Boom in 2018. [WWW] Saadaval: https://www.inc.com/adam-heitzman/8-innovative-industries-that-will-really-boom-in-2018.html. (12.04.2018)

[3] Epp-Mare Kukemelk. 2013. 21 pilootkooli õpilased hakkavad programmeerimist õppima. [WWW] Saadaval: http://www.delfi.ee/news/paevauudised/eesti/21-pilootkooli-opilased-hakkavad-programmeerimist-oppima?id=65982524. (12.04.2018).

[4] Katrina Owen. 2017. Towards an Exercism Roadmap. [WWW] Saadaval: https://github.com/exercism/discussions/issues/113 (17.05.2018)

[5] Oleg Shelajev. 2017. Java Web Frameworks Index: February 2017. [WWW] Saadaval: https://zeroturnaround.com/rebellabs/Java-web-frameworks-index-by-rebellabs/. (13.04.2018)

[6] Comparison with Other Frameworks. [WWW] https://vuejs.org/v2/guide/comparison.html (13.04.2018)

[7] DB-Engines Ranking [WWW] https://db-engines.com/en/ranking (13.04.2018)

[8] AG10. 2018. Docker. Начало. [WWW] Saadaval: https://habr.com/post/353238/. (24.04.2018)

[9] Vincent Driessen. 2010. A successful Git branching model. [WWW] Saadaval: http://nvie.com/posts/a-successful-git-branching-model/. (14.04.2018)

LISAD

Lisa 1. Programmeerimisülesannete agregeerimise tulemusena tekkinud ülevaade ja statistika

Lihtsad (3 punkti):

Ülesanne	Kategooria	Inimest eksamil/ 0p saanud	Keskmine %	Pärineb	Kasutatud platvormil
ReverseEnd	Tsükkel	24/0	100%	E2017	+
SumNot42	Tsükkel, massiiv	44/1	98%	E2017	+
AlignRight	Tsükkel, sõne	49/1	98%	E2017	+
SumString	Sõne	44/1	96%	E2017	+
SumPlusMinus	Tsükkel, moodul, massiiv	49/3	94%	E2017	+
MaxOdd	Tsükkel, moodul, massiiv	42/2	91%	E2017	+
Average	Sõne, matemaatika	49/3	91%	E2017	+
RecReverse	Rekursioon	13/1	90%	E2015	
DoubleElements	Tsükkel, massiiv	44/5	88%	E2017	+
SmallerThan	Sõne	42/5	88%	E2017	+
AddIndex	Tsükkel, massiiv	49/6	88%	E2017	+
Diff	Sõne, matemaatika	24/2	88%	E2017	+
AddReverse	Tsükkel, sõne	42/3	86%	E2017	+
LastEven	Tsükkel, massiiv, moodul	24/1	85%	E2017	
PartialSum	Massiiv	3/0	83%	E2016	
MaxDistance	Massiiv	13/1	81%	E2016	
CapitalizeWords	Sõne	14/0	80%	E2016	+
SameEndWords	Järjend, sõne	42/4	79%	E2017	
Split	Järjend	49/8	77%	E2017	
BinaryToDecimal	Algoritm, massiiv	14/0	75%	E2016	
CountElement	Massiiv	42/10	75%	E2017	
GenerationZ	Järjend, regex	24/1	75%	E2017	
DoubleReverse	Tsükkel, massiiv	24/4	72%	E2017	
LowerEverySecond	Tsükkel, sõne, moodul	44/12	64%	E2017	
SimilarWords	Sõne, järjend	44/13	64%	E2017	

Keskmised (5 punkti):

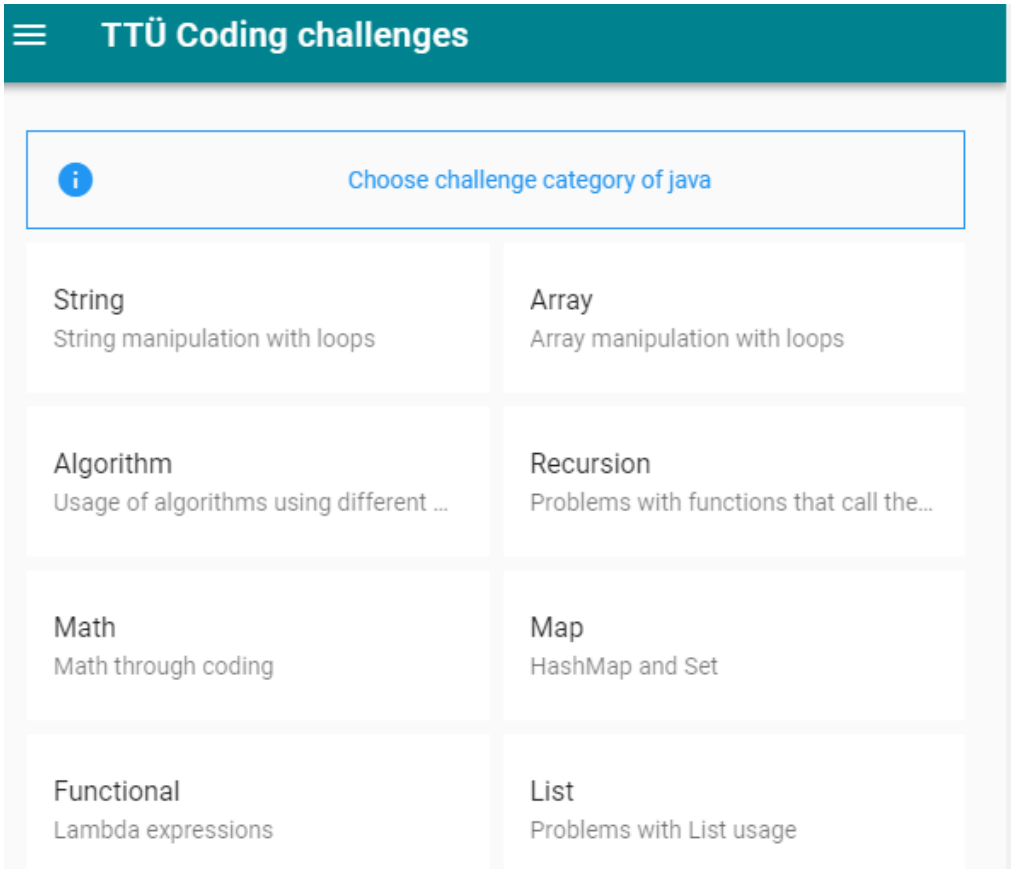
Ülesanne	Kategooria	Inimest eksamil/0p saanud	Keskmine %	Pärineb	Kasutatud platvormil
RecSymbol	Rekursioon	14/0	86%	E2016	+
ReplacePair	Rekursioon	11/1	86%	E2015	
RecPower	Rekursioon	14/0	84%	E2016	+
Calculate	Algoritm, matemaatika	6/0	82%	E2016	+
FindRecSubarrayCount	Rekursioon	23/1	82%	E2015	
IsCorrectParentheses	Sõne, algoritm	23/0	81%	E2015	
Anagram	Sõne, algoritm, HashMap	3/0	80%	E2016	+
RecMax	Rekursioon	11/0	80%	E2016	+
RecUnpack	Rekursioon	3/0	77%	E2016	
RecCenter	Rekursioon	14/0	76%	E2016	
OneStarOut	Rekursioon	44/7	76%	E2017	
RecGrowth	Rekursioon	49/9	76%	E2017	
SplitNumbers	Massiiv, sõne, järjend	14/2	75%	E2016	
CountOfSums	Algoritm, massiiv	6/0	73%	E2016	+
Skip	Rekursioon	6/0	71%	E2016	+
TimeDiff	Algoritm, Sõne	11/0	71%	E2016	+
Stack	Stack realiseerimine OOP	44/7	71%	E2017	
MinimizeText	Sõne	14/0	70%	E2016	+
PhoneBook	OOP	42/12	70%	E2017	
RemainingChair	Massiiv, algoritm, matemaatika	14/0	69%	E2016	+
RecCount	Rekursioon	42/5	69%	E2017	
FindMissingNumber	Massiiv	14/0	67%	E2016	+
RecSeparate	Rekursioon, sõne	24/8	66%	E2017	
LongestSymSubArray	Matemaatika, massiiv	19/0	66%	E2015	
RecDoubleReverse	Rekursioon	10/1	63%	E2016	
TrackDurationSum	Matemaatika, järjend, massiiv	49/11	62%	E2017	+
RecVowel	Rekursioon, raskem	10/0	60%	E2016	
Height	Rekursioon	14/1	60%	E2015	
HalfSum	Tsüklid, massiiv, algoritm	11/1	59%	E2016	+

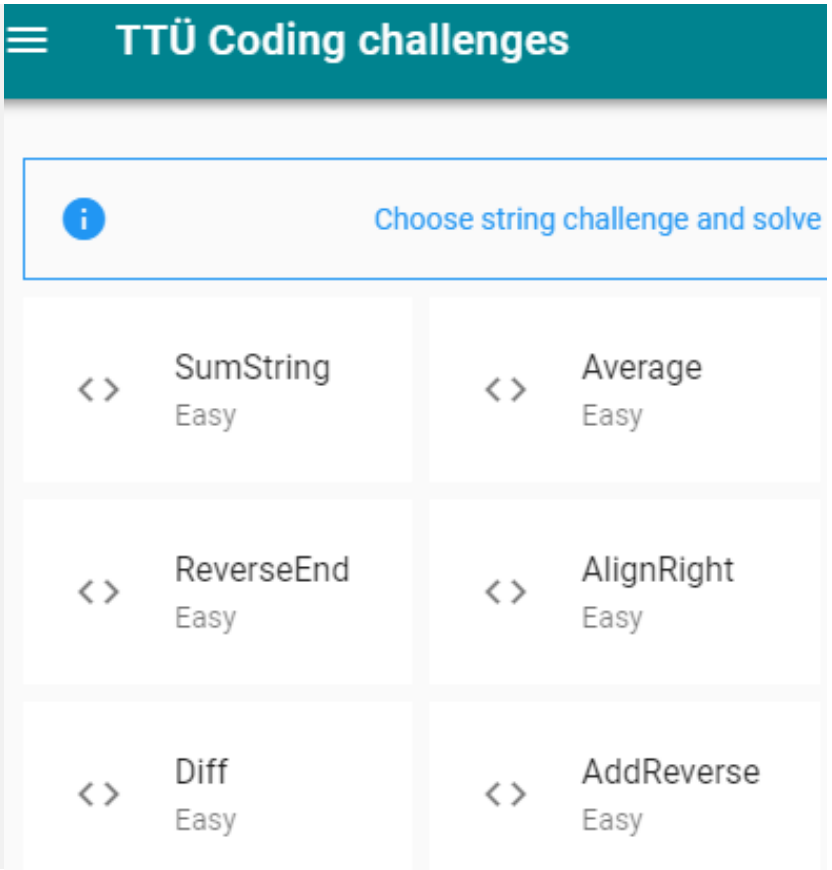
Order	OOP, järjend	49/10	59%	E2017	
SumInArray	Masiiiv, Matemaatika	11/1	58%	E2015	
MixedPairs	ASCII, sõne	10/0	57%	E2016	
FindClosest	Algoritm, massiiv, matemaatika	12/7	57%	E2016	+
Collatz	Matemaatika, massiiv, moodul	17/3	56%	E2016	+
CountChars	Sõne, HashMap	11/1	56%	E2015	
CombineSortedArrays	Massiiv	23/2	55%	E2015	
SecondLargestOddNumber	Massiiv, moodul	14/0	55%	E2016	+
RecChars	Rekursioon, raskem	14/2	55%	E2016	
PaperScissorsRock	Tsükkel, massiiv	42/11	55%	E2017	
SumNumbers	Algoritm, sõne, järjend	12/2	55%	E2016	+
ReverseWordsWithSpaces	Sõne	15/2	54%	E2015	
Move	Massiiv	44/14	54%	E2017	
ConsistsCount	Massiiv	10/2	53%	E2016	
Quotient	Matemaatika, järjend, massiiv	10/0	53%	E2016	+
FindSub	Algoritm, massiiv	11/2	53%	E2016	+
RecAsymmetryCount	Rekursioon, raskem	17/2	50%	E2016	+
CarShop	OOP	24/9	45%	E2017	
FindOp	Massiiv	13/3	45%	E2015	
IsAsc	Sõne, matemaatika	19/0	42%	E2015	
RecSumNumber	Rekursioon, palju raksem	14/1	40%	E2016	
Transactions	Massiiv	24/12	32%	E2017	

Rasked (10 punkti):

Ülesanne	Kategooria	Inimest eksamil/0p saanud	Keskmine %	Pärineb	Kasutatud platvormil
FindWordBothSides	Sõne, algoritm, HashMap	10/1	60%	E2016	+
MaxSubstring	Sõne, algoritm	7/0	51%	E2016	+
MaxSubCount	HashMap, algoritm	49/25	40%	E2017	+
NestedParent	Rekursioon, Algoritm	15/1	43%	E2015	
CombineNumbers	Massiiv, matemaatika, algoritm	7/2	34%	E2016	+
MaxGrowth	Massiiv, Algoritm	44/15	34%	E2017	
Alphabet	Sõne, tähestik ASCII	17/5	29%	E2016	
RecHasEvenTwos	Rekursioon, Raskem	11/3	26%	E2016	+
DeepestParentContent	Algoritm, sõne	7/4	26%	E2016	+
MaxDiffSub	Massiiv	18/5	25%	E2015	
FindWord	Algoritm	42/24	23%	E2017	
Combine	Algoritm	24/6	19%	E2017	
TicTacToe	OOP, Matemaatika, Algoritm	49/35	17%	E2017	
University	OOP	42/26	16%	E2017	
Expression	Matemaatika, algoritm	11/5	15%	E2016	+
Booking	OOP	44/27	15%	E2017	
Album	OOP, keerulisem	24/16	8%	E2017	

Lisa 2. Kasutajaliidese kategooria ja ülesande valiku vaade





Lisa 3. Kasutajaliidese ülesande ja testimissüsteemist saadud tulemuse vaade