TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Triinu Malv 193654IADB

Klientrakenduse loomine algoritmilist mõtlemist arendavale TalTechi veebimängule tALGOtech

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Abijuhendaja: Kerman Saapar
Bakalaureusekraad

Tallinn 2022

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: Triinu Malv

15.05.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua töötav klientrakendus algoritmilist mõtlemist arendavale Tallinna Tehnikaülikooli veebimängule tALGOtech, mille abil viia läbi praktilisi töötube IT Kolledži erialasid tutvustavatel ekskursioonidel. Veebimängu peamiseks sihiks on äratada potentsiaalsetes tudengites huvi algoritmilise mõtlemise ja infotehnoloogia õppimise vastu. Klientrakendus on Tallinna Tehnikaülikooli töötajatele ja üliõpilastele avatud lähtekoodiga ning selle loomisel on silmas peetud projekti paindlikkust, tagamaks võimalikult mugavaid tingimusi edasiarenduseks.

Arendusprotsessi käigus luuakse klientrakendus, mis võimaldab kasutajal etteantud vahendeid õigesse järjekorda nihutades ülesande lahendamiseks algoritm koostada. Pärast algoritmi käivitamist genereerib klientrakendus serverrakendusest saadavate andmete põhjal käimasolevale mängule ja kasutaja algoritmile vastava animatsiooni. See näitab kasutajale tema antud juhiste järgi tehtavaid tegevusi ning aitab seeläbi lahendusest veakohad üles leida. Arenduse peamisteks osadeks on algoritmi koostamise võimaluse loomine ning sisendist sõltuva animatsiooni teostamine.

Töö keskendub olemasolevate alternatiivide puudusi katva lahenduse loomisele, kasutades kaasaegseid, suure kasutajabaasiga ning rakenduse vajadustele vastavaid tehnoloogiaid. Fikseeritakse nõuded, analüüsitakse erinevaid tehnoloogilisi võimalusi ning antakse ülevaade arendusprotsessist, mille tulemuseks on töötav klientrakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 43 leheküljel, 6 peatükki, 22 joonist, 3 tabelit.

Abstract

Creating a Front-end Application for TalTech's Computational Thinking Web Game tALGOtech

The aim of the current thesis is to create a front-end application for TalTech's computational thinking web game tALGOtech, that would help conduct workshops during IT College's programmes introductory tours. The main purpose of this web game is to arouse interest against algorithmic thinking and studying IT. The application's source code is open for the employees and students of TalTech and the project is designed considering possible future developments.

During the development process, a front-end application is created that enables the user to compose an algorithm by dragging and dropping given elements into a suitable sequence. After executing the algorithm, the application generates an animation by the data it receives from the back-end. The animation shows the user what their algorithm does and helps them easily find the points of failure. The main parts of the development process are creating the functionality for algorithm-composing and implementing dynamic animation that depends on the input.

Current thesis focuses on creating a solution that would cover the deficits of existing alternatives using modern and popular technologies that would be suitable for the application's needs. Requirements are set, different technological options are analyzed and an overview of the development process is given, which results in a functioning front-end application.

The thesis is in Estonian and contains 43 pages of text, 6 chapters, 22 figures, 3 tables.

Lühendite ja mõistete sõnastik

AJAX	*Asynchronous JavaScript and XML*, asünkroonseid päringuid võimaldav tehnoloogiate kogum
CRUD	*Create, Read, Update, Delete*, arvutiprogrammide püsiva salvestamise neli põhitoimingut – loomine, lugemine, värskendamine ja kustutamine
CSS	*Cascading Style Sheets*, küljendamisel kasutatav märgistuskeel
DOM	*Document Object Model*, dokumendi objektimudel – platvormist ja keelest sõltumatu dokumentidega suhtlemise liides
HTML	*HyperText Markup Language*, veebilehe märgendamise keel
HTTP	*HyperText Transfer Protocol*, protokoll teabe edastamiseks arvutivõrkudes
IT	*information technology*, infotehnoloogia
JSON	*JavaScript Object Notation*, JavaScripti programmeerimiskeele alamhulgal põhinev lihtsustatud andmevahetusvorming
JWT	*JSON Web Token*, internetistandard andmete loomiseks koos krüpteerimisega
Klientrakendus	Rakendus, mis liidestab kasutajat süsteemi tagaosaga
MPA	*Multi Page Application*
NPM	*Node Package Manager*, JavaScripti paketihaldur
SPA	*Single Page Application*
SVG	*Scalable Vector Graphics*, XML-il põhinev keele spetsifikatsioonide hulk, mis kirjeldab kahemõõtmelist vektorgraafikat
XML	*Extensible Markup Language*, standardne üldotstarbeline märgistustkeel

Sisukord

1 Sissejuhatus 10
2 Ülesandepüstitus 11
2.1 Metoodika 11
2.2 Algoritmiline mõtlemine 11
2.2.1 Algoritmiline mõtlemine igapäevaelus 12
2.2.2 Algoritmilise mõtlemise õppimine 12
2.3 Olemasolevad lahendused 13
2.3.1 Scratch 13
2.3.2 Hopscotch 13
2.3.3 The Pack 14
2.3.4 Autothinking 14
2.4 Töö kitsendused 15
2.5 Pakutav lahendus 15
3 Loodava veebirakenduse analüüs 17
3.1 Nõuete määramine 17
3.1.1 Funktsionaalsed nõuded 17
3.1.2 Mittefunktsionaalsed nõuded 18
3.1.3 Tulevikus loodavate funktsionaalsuste nõuded 19
3.2 Tehnoloogia valik 20
3.2.1 Serveripoolse rakenduse tehnoloogiate valik 20
3.2.2 Klientrakenduse raamistik 21
3.2.3 Klientrakenduse disainiraamistik 23
3.2.4 Klientrakenduse animeerimisraamistik 24
3.3 Arenduskeskkonna valik 26
3.3.1 Koodihalduse keskkonna valik 26
3.3.2 Integreeritud arenduskeskkonna valik 27
3.4 Veebirakenduse disain 28
3.4.1 Kasutajakogemuse disain 28
3.4.2 Kasutajakogemuse disain Figma tarkvara abil 30

3.4.3 Andmebaasi mudel 32
3.5 Analüüsi kokkuvõte 33
4 Klientrakenduse arendus 35
4.1 Komponentide planeerimine 35
4.2 React – *Single Page Application (SPA)* 37
4.3 Rakenduse struktuur 37
4.4 MUI disainiraamistiku kasutus 38
4.5 Suhtlus veebiteenusega 39
4.5.1 Päringute tegemine klientrakenduses 39
4.5.2 Ühenduspunktid ja vahetatavad andmekujud 40
4.6 Lahendusalgoritmi koostamise võimaluse loomine React DnD abil 43
4.7 Sisendile vastava animatsiooni loomine Anime.js teegi abil 46
4.8 Raskusastmete erisused kasutajaliideses 48
4.9 Serverrakenduse arendus 49
5 Hinnang loodud klientrakendusele 50
5.1 Kasutatavus 50
5.2 Kasutatud tehnoloogiate uudsus ja sobivus 51
5.3 Võimalused edasiseks arenduseks 51
6 Kokkuvõte 53
Kasutatud kirjandus 54
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 58
Lisa 2 – tALGOtech infosüsteemi skeem 59
Lisa 3 – Andmebaasi täielik olemi-suhte diagramm 60
Lisa 4 – tALGOtech kasutusjuhend töötoa läbiviijale 60
Lisa 5 – tALGOtech klientrakenduse koodihoidla link 65
Lisa 6 – tALGOtech rakenduse link 65

Jooniste loetelu

Joonis 1. Tavakasutaja ülesande lahendamise kasutajakogemuse diagramm. 29
Joonis 2. Administraatori kasutajakogemuse diagramm. 29
Joonis 3. Tavakasutaja ülesande lahendamise vaate prototüüp enne lahendamist. 31
Joonis 4. Tavakasutaja ülesande lahendamise vaate prototüüp koos lahendusega. 31
Joonis 5. Tavakasutaja vaate prototüüp loodud algoritmi animatsiooni vaatamiseks. ... 32
Joonis 6. Lihtsustatud olemi-suhte diagramm. 33
Joonis 7. Valitud mängu ülesande lahendamise alamlehe React komponentide paiknemine. 36
Joonis 8. tALGOtech klientrakenduse projekti struktuur. 38
Joonis 9. Kasutaja loodud lahenduse saatmise päring animatsiooni komponendis. 39
Joonis 10. POST päring päringute tegemiseks loodud baasklassis. 40
Joonis 11. Iga mängu põhilise info liides. 41
Joonis 12. Valitud mängu kogu info liides. 41
Joonis 13. Vajaliku päringu tuvastamine ja teostamine mängu laadimiseks. 42
Joonis 14. Lahenduskasti liides IBox. 42
Joonis 15. Kasutaja lahenduse tulemust iseloomustav liides ISolutionResult. 43
Joonis 16. Lüngatäite komponendi hiirega pukseerimise loogika React DnD abil. 44
Joonis 17. Pillatava kasti vastuvõtmine lahenduse komponendis React DnD abil. 45
Joonis 18. Fiktiivne näidisalgoritm ülesande lahendamise vaates. 46
Joonis 19. Animeeritavate pildielementide genereerimine vastavalt sisendile. 46
Joonis 20. Tegevuste ja kirjeldavate tekstide käitumise lisamine animatsiooni ajajoonele. 47
Joonis 21. Animatsioonivaade puu istutamisest. 48
Joonis 22. Pooleliolev lahendus mängu kergel tasemel. 49

Tabelite loetelu

Tabel 1. Klientrakenduse raamistike võrdlus. 22
Tabel 2. Klientrakenduse disainiraamistike võrdlus..... 24
Tabel 3. Klientrakenduse animeerimisraamistike võrdlus. 25

1 Sissejuhatus

Käesoleva lõputöö raames luuakse Tallinna Tehnikaülikooli IT Kolledžile klientrakendus algoritmilise mõtlemise töötubade läbiviimiseks. Eesmärgiks on teostada kasutusvalmis rakendus, mis aitaks äratada huvi algoritmilise mõtlemise vastu ning õpetada seda probleemide lahendamisel kasutama. Klientrakenduse kasutatav serverrakendus valmib kaastudeng Maarja Helena Elisabeth Hoopi bakalaureusetööna "TalTechi algoritmilise mõtlemise veebimängu serverrakenduse arendus".

Vajaduse uudse veebirakenduse järele tekitas eelkõige olemasolevate alternatiivide sobimatus nende töötoa formaadis kasutamiseks. Kuna peamiselt on vajaminev rakendus mõeldud esmakordsetele kasutajatele ning teatud piiratud ajavahemikus, on tähtis kasutajaliidese intuitiivsus ning madal õppimislävi ülesande lahendamise vahendite kasutamiseks. Olemasolevad alternatiivid nõuavad põhjalikku keskkonnaga tutvumist, mis ei võimalda keskenduda töötoa tegelikule eesmärgile.

Lahendusena pakutakse klientrakendust, mis on üles ehitatud, pidades silmas algoritmilise mõtlemise alustalasid, loogiliste konstruktsioonide keeleliselt väljendamise põhimõtteid, kaasaegsete tehnoloogiate kasutamist ning sujuva ja intuitiivse kasutajakogemuse loomist. Kasutaja arengu tagamiseks kohandatakse esitatava ülesande raskusastet vastavalt tema toimetulekule, mis aitab saavutada eduelamust ja alustada algoritmilise mõtlemisega tutvumist samm-sammult vastavalt igaühe isiklikele võimetele.

2 Ülesandepüstitus

Eestis on tänasel tööturul programmeerijad kõrges hinnas. Paljuski tuleneb see ka vähesest arendajate pealekasvust. Selleks, et osata erialavalikul just IT kasuks otsustada, on oluline tutvustada mõttemudelit, mida selles maailmas probleemide lahendamiseks vaja läheb – algoritmilist mõtlemist.

2.1 Metoodika

Sobiva lahenduse väljatöötamiseks kirjeldatakse esmalt käsitletavat probleemi, seda puudutava valdkonna olulisemaid aspekte ning juba olemasolevaid alternatiivseid lahendusi. Selle tulemusena pakutakse välja probleemi nõuetele vastav IT lahendus ning pannakse paika, mis kuulub käesoleva lõputöö skoopi ja mis jääb tulevikku edasiarenduseks.

Lahenduse analüüsi käigus tehakse kindlaks sellele seatavad funktsionaalsed ja mittefunktsionaalsed nõuded. Seejärel vaadeldakse erinevaid võimalikke tehnoloogilisi vahendeid, mille hulgast valitakse arenduseks sobivaimad, et jõuda soovitud tulemuseni. Sujuvama arendusprotsessi tagamiseks ning senini tähelepanuta jäänud kitsaskohtade leidmiseks luuakse ka prototüüp.

Arendusprotsessi kirjelduses keskendutakse ülevaate andmisele lahenduse tähtsamate osade tööpõhimõtetest ja nii klientrakenduse siseselt kui läbi serverrakenduse ühenduspunktide liikuvatest andmetest. Käesoleva töö lõpuosas kirjeldatakse valminud lahenduse tulemusi ning edasiarendusvõimalusi.

2.2 Algoritmiline mõtlemine

Algoritmiline mõtlemine on lähenemisviis probleemile lahenduse leidmiseks, mis põhineb probleemi eri osadeks tükeldamisel, nende hulgast sarnasuste ja lihtsustamisvõimaluste otsimisel ning seejärel eelnevat arvesse võttes lahenduse loomisel. Täpsemalt hõlmab see lahendustehnika nelja võtmeelementi: probleemi

lõhustamine, mustrite tuvastamine, abstraktsioon ja algoritmid. Lõhustamine võimaldab väiksemateks tükkideks jagatud probleemide lahendusi kombineerides lahendada esialgne keeruline väljakutse. Mustrite tuvastamine on oluline loodava lahenduse lihtsustamiseks: otsitakse sarnasusi varasemalt lahendatud probleemidega ning vaadeldakse, kas probleemisiseselt esineb teatud kordusi. Abstraktsioon ehk üldistamine nõuab probleemi vaatlemist ilma spetsiifilisi detaile arvesse võtmata, mis võimaldab luua üldisema, rohkemaid probleeme katva lahenduse. Viimaks tuleb luua ideeliselt väljamõeldud lahenduse teostamiseks samm-sammuline plaan ehk algoritm, mida on võimalik kirja panna nii tavatekstina, joonisena kui pseudokoodina [1].

2.2.1 Algoritmiline mõtlemine igapäevaelus

Vaatamata töös kasutatavale lähenemisele vaadelda algoritmilist mõtlemist kui midagi, mida tuleks eraldi õppida, on siiski tegu märkamatult igapäevaelus rakendatava metoodikaga. Loogiline mõtlemine on inimese intellekti juures üks loomulikest valdkondadest, kuid tavaliselt ei ole erinevate mõtlemisviiside vahel selgelt tajutavaid piire. Sellist esmapilgul küllaltki ähmast valdkonda on aga võimalik täpselt esitada kindlate reeglite alusel koostatud tekstide abil [2, p. 7].

Nõudes valdkonnavõõralt inimeselt intuitiivselt mõistetava struktuuriga loogiliste lausete moodustamist, on võimalik suunata teda oma algoritmilist mõtlemist arendama, kasutades ära tema juba niigi igapäevaselt kasutusel olevaid oskusi.

2.2.2 Algoritmilise mõtlemise õppimine

On täheldatud, et õpilaste edukus on tugevas seoses nende võimega rakendada probleemide ületamiseks algoritmilise mõtlemise tehnikaid. Selle arendamiseks soovitatakse kasutada ülesandeid, mida on keeruline lahendada, kuid mis muutuvad ümberdefineerimisel ja visualiseerimisel kergemini mõistetavaks. Samuti tuleb algoritmide paremini mõistma õppimisel kasuks nende graafiline või animeeritud kujutamine [3].

Ehkki algoritmiline mõtlemine on üks olulisemaid eduka arendaja tööriistu, pole selle õppimiseks programmeerimisoskust tarvis. Algoritmi loomine eeldab samm-sammuliste juhiste koostamist, mida võib vaadelda ka probleemi lahendamise täpsete reeglitena. Seetõttu on hea viis tutvustada algoritmilise mõtlemise abil probleemide lahendamist läbi

selleks mõeldud mängude – mängimine seisnebki enamasti teatud reeglistikku järgides vajalike sammude läbimises [4].

2.3 Olemasolevad lahendused

Olemasolevate lahenduste valikusse võeti mängud, mis määratlevad end algoritmilise mõtlemise või algelise programmeerimisoskuse arendamise vahenditena. Vaadeldi nii nende funktsionaalsuse kui kasutajaliidese vastavust lõputöö rakenduse vajadustele. Hinnang kujundati mängude kodulehtedelt kogutud informatsiooni ning isikliku katsetamise kombinatsioonina.

2.3.1 Scratch

Scratch on populaarne ning lihtsastimõistetava visuaalse liidesega programmeerimiskeel, millega saab luua mänge ja animatsioone. Peamiselt on see suunatud 8-16-aastastele lastele, kuid on heaks esimese programmeerimisaimduse saamise abivahendiks kõigile. Scratch'i eesmärgiks on edendada arvutuslikku mõtlemist ja probleemide lahendamise oskusi, millele lisaks peetakse tähtsaks ka loova eneseväljenduse arendamist [5]. Olemas on põhjalikud materjalid, mis juhendavad õpetajaid Scratch'i abil tunde läbi viima ning laiemalt arvutuslikku mõtlemist tutvustama [6]. Lisaks on kasutajatel võimalik enda loodud mänge avalikustada, mida on seejärel võimalik juba kõigil proovida [7].

Scratch'i puuduseks sellega esmakordselt kokku puutuvatele inimestele töötoa korraldamiseks on selle vahendite rohkus ning keerukus. Kuigi lai funktsionaalsus ja suur hulk tööriistu annavad võimaluse luua unikaalsema lõpptulemuse, kujuneb see takistuseks, kui olulisem on aja piiratuse tõttu võimalikult kiiresti ülesande sisulisele lahendusele keskenduma hakata.

2.3.2 Hopscotch

Hopscotch on eelkõige 9-15-aastastele lastele loodud mobiilirakendus programmeerimise õppimiseks. Sellega saab teha nii mänge, animatsioone kui oma rakendusi. Olemas on ka võimalus oma loomingut teiste kasutajatega jagada ning ise teiste projekte täiendada. Hopscotch pakub suurt valikut videoõpetusi, mis aitavad ise luua oma versiooni mitmetest tuntud mobiilimängudest nagu Pokemon Go ja Geometry Dash [8].

Töö eesmärkide jaoks sobimatuks muudab Hopscotch'i taas peamiselt selle keerukus. Rakenduse ülesehituse ning võimalustega tutvumine nõuaks selle töötoa käigus kasutamisel sisulise tööga võrreldes ebaproportsionaalselt palju aega. Samuti on see mõeldud peamiselt ise millegi tegemiseks, mitte niivõrd etteantud ülesandele lahenduse otsimiseks. Olulise puudusena võib välja tuua ka selle kättesaadavuse – rakendusele ligi pääsemiseks tuleb see endale telefoni või tahvelarvutisse alla laadida, mida saavad omakorda teha vaid Apple'i seadmete kasutajad. Lisaks on mõned funktsionaalsused kättesaadavad vaid kuutasu eest.

2.3.3 The Pack

The Pack on mäng, kus kasutaja eesmärgiks on otsida toitu ja vett ning selle käigus erinevate funktsioonidega olendeid leides oma salka suurendada. Kui seltskonda on kogunenud vähemalt kaht tüüpi olendeid, saab mängu edukamaks läbimiseks luua olendite funktsionaalsustest algoritme. Mängu arenedes ning enamaid olenditüüpe kaasates on võimalik luua ka üha keerulisemaid ning kasulikumaid algoritme [9].

Ehkki eelisena võib näha mängu suurt sarnasust muude mobiilimängude mängimise protsessiga, on tegu eelkõige pikaajaliseks kasutamiseks mõeldud lahendusega. Esmakasutaja jaoks on keeruline ilma lisajuhiseid saamata mängu eesmärki ning selleni jõudmise võimalusi mõista. Samuti kulub küllaltki kaua aega, enne kui mäng algoritmiliselt keerukate probleemide lahendamiseni jõuab. Puuduseks on ka vajadus rakendus alla laadida, kuna sellel pole veebiversiooni.

2.3.4 Autothinking

Autothinking on algoritmilise mõtlemise arendamiseks mõeldud mäng, mis õpetab kasutaja tegevustega kohandudes algoritmilise mõtlemise kasutamist ning selle mõisteid. See on välja töötatud eeskätt lasteaialastele ja kooliõpilastele ning kasutab lahenduse loomiseks teksti või programmeerimiskäskude asemel ikoone [10].

See lahendus on oma põhiolemuselt töö eesmärkidega küllaltki sarnane, kuid kasutamise käigus tulevad siiski välja mitmed puudused. Oluliseks murekohaks on mängijale antava ülesande tükeldamine. Hiir peab suures labürindis liikudes kõik juustud kätte saama, seejuures kassiga kokku sattumata. Rakendus võimaldab aga korraga maksimaalselt kümne sammu kirjapanekut, misjärel loodud jada kaob ning mängija alustab uue loomist. Suureks eeliseks olev kohanduvus on loodud vaid mängu kolmandale tasemele, millele

ligipääsemiseks tuleb mängu loojatega isiklikult ühendust võtta. Mängu graafiline lahendus on küllaltki algeline ning lahenduse loomiseks kasutatavad ikoonid pole intuitiivselt mõistetavad.

2.4 Töö kitsendused

Loodava rakenduse fookus on kitsendatud vastama eelkõige selle tellija, Tallinna Tehnikaülikooli IT-teaduskonna, vajadustele. Tähtis on rakenduse intuitiivsus ning selle kasutamise madal õppimislävend, kuna seda plaanitakse rakendada ülikooli IT-erialasid tutvustavate ringkäikude praktilise osana. Keskkonna ning selle vahendite tundmaõppimise asemel peab lahendaja võimalikult kiiresti saama keskenduda tegelikule ülesandele. Peamiselt on potentsiaalsete kasutajatena arvestatud põhikooli lõpuosa või gümnaasiumiastme esimese poole õpilasi, kellel seisab lähiajal ees erialavaliku langetamine. Eesmärgiks on anda neile aimu algoritmilise mõtlemise olemusest ning aidata seeläbi infotehnoloogia rohkemate inimeste erialavalikute hulka.

2.5 Pakutav lahendus

Olemasolevate lahenduste suurimaks läbivaks puuduseks on nende suur õppimiskeerukus, mis takistab nende rakendamist töötoa formaadis, eriti kui enamjaolt on osalejate näol tegu süsteemi esmakasutajatega. Probleem on ka rakenduste mugava kättesaadavusega. Scratch ja Hopscotch pole suunatud mitte niivõrd etteantud ülesannete lahendamisele, kui pigem ise millegi täiesti uue loomisele. Seeläbi õpitakse küll erinevaid programmeerimises kasutatavaid elemente, kuid mitte algoritmilise mõtlemise rakendamist probleemide lahendamiseks. Autothinking, ehkki raskesti ligipääsetav, võib küll olla heaks vahendiks esmaseks algoritmidega tutvumiseks, kuid jääb pigem siiski väiksemate laste tasemele ning pole kuigi intuitiivselt kasutatav, vajades eelnevalt tutvumist süsteemi küllaltki pika õpetusvideoga.

Vajadusi katva lahendusena pakutakse välja uus veebipõhine rakendus tALGOtech, mis esitaks kasutajatele lahendamiseks programmeerimises esinevate väljakutsetega olemuselt sarnaseid probleeme. Tähtis on ülesande olemuse kerge mõistetavus, kuid sellele lahenduse väljamõtlemise keerukus. Töötoa vormis kasutamiseks on oluline, et esmakasutaja saab võimalikult kiiresti rakendust kasutama asuda. Õppimiskeerukuse vähendamiseks tuleb tagada eelkõige ülesande lahendamiseks etteantud vahendite

kasutamise intuitiivsus. Nendest loodav algoritm peaks tekkima keeleliselt loogiliste tükkide omavahelisel ühendamisel. Seeläbi moodustuvad argikeelele sarnanevad laused, tänu millele on kasutajal kergem mõista algoritmi koostamise põhimõtteid ning selle toimimist. Erinevatel tasemetel mängijatele võimalikult hea kasutajakogemuse pakkumiseks kohandab tALGOtech raskusastet vastavalt mängija tasemele. Selleks muudetakse nii ülesande sisu, mis reguleerib lahendamiseks vajaminevate algoritmiliste konstruktsioonide keerukust, kui kohandatakse etteantavate vahendite hulka. Nii näeb kasutaja kergel tasemel täpselt vajaminevat arvu vahendeid, samas kui keskmisel ja raskel tasemel pole teada, mitu korda vahendeid eduka lahenduse koostamiseks kasutada tuleb. Olulise lisandusena luuakse ka võimalus näha loodud algoritmi toimimist animatsiooni abil, mille lõppedes kuvatakse kasutajale ka tema lahenduse tulemus koos vihjega selle parandamiseks.

Lõputöö käigus loodav lahendus keskendub vaid ühe ülesande lahendamiskeskkonna kasutajaliidese ning selle algoritmi kujutava animatsiooni loomisele, kuid süsteemi planeerimise käigus arvestatakse ka tulevikus loodavate lisafunktsionaalsuste mugava edasiarendamise võimaldamisega. Tähtsamate edasiarendustena nähakse kasutajate süsteemi teostamist ning nendepoolsete mängude loomise võimaluse tekitamist.

Kasutajaliidese loomisel piirdutakse brauseripõhise lahendusega, kuna rakendus on mõeldud kasutamiseks eelkõige klassiruumides.

Töö raames valmiva rakenduse eesmärk pole raha teenimine, vaid Tallinna Tehnikaülikoolile IT-erialasid tutvustava praktilise vahendi loomine. tALGOtechi projekti kood jääb Tallinna Tehnikaülikooli tudengitele ja töötajatele avalikuks, tänu millele saavad üliõpilased seda tulevikus kergesti edasi arendada.

3 Loodava veebirakenduse analüüs

Analüüsi käigus selgitatakse välja täpsed nõuded ning valitakse tehnoloogiad lahenduse teostamiseks. Lisaks disainitakse eeldatav kasutajakogemus ning luuakse visuaalne prototüüp.

3.1 Nõuete määramine

Nõuded on loodud Tallinna Tehnikaülikooli – rakenduse tellija soovide ning süsteemi autorite omapoolsete täienduste kombineerimisel. Nõuete määramisel on arvesse võetud, et rakendus käsitleb algoritmilise mõtlemise tutvustamise probleemi Eesti ulatuses, olles suunatud peamiselt põhikooli- ning gümnaasiumiastme noortele.

Nõuded põhinevad kasutajalugudel, mille gruppideks on endas algoritmilise mõtlemise oskust arendada sooviv registreerimata kasutaja (edaspidi "tavakasutaja") ning veebirakendust haldav ja seal olevaid ülesandeid ajakohasena hoidev administraator.

Lisas 2 on kujutatud järgnevalt kirjeldatud nõuete põhjal loodud infosüsteemi skeem.

3.1.1 Funktsionaalsed nõuded

Tavakasutaja funktsionaalsed nõuded:

- Tavakasutajana soovin valida endale huvipakkuva temaatikaga mängu.
- Tavakasutajana soovin analüüsida püstitatud ülesannet ning selle lahendamiseks etteantud vahendeid.
- Tavakasutajana soovin lohistada programmi koostamiseks mõeldud kastid endale sobivana näivasse järjekorda.
- Tavakasutajana soovin täita programmiks lohistatud kastide lüngad selleks ettenähtud fraaside või numbritega.
- Tavakasutajana soovin juba programmis olevaid kaste ümber järjestada.
- Tavakasutajana soovin koostatud programmist valitud kaste ja lüngatäiteid kustutada.

- Tavakasutajana soovin teatud tegevusi paigutada tingimuslause või iteratsiooni kasti mõjualasse, lohistades need vastava kasti lisaalasse.
- Tavakasutajana soovin käivitada enda loodud ülesande lahenduse.
- Tavakasutajana soovin graafiliselt näha enda loodud programmi toimimist.
- Tavakasutajana soovin enda loodud programmi muuta, kui see ülesande tingimusi ei täida.
- Tavakasutajana soovin korduva ebaõnnestumise korral valida lihtsama raskusastmega ülesande lahendamise.
- Tavakasutajana soovin lihtsamal tasemel ülesande lahendamiseks näha täpselt vajalikku hulka etteantud vahendeid.
- Tavakasutajana soovin õnnestumise korral valida raskema raskusastmega ülesande lahendamise.
- Tavakasutajana soovin keskmisel ja raskel tasemel ülesande lahendamiseks näha vahendeid, mille hulgas esineb mitmekordset kasutusvõimalust.

Administraatori funktsionaalsed nõuded:

- Administraatorina soovin sisse ja välja logida.
- Administraatorina soovin mänge lisada.
- Administraatorina soovin mänge muuta.
- Administraatorina soovin mänge kustutada.
- Administraatorina soovin määrata kasutajale igal raskusastmel lubatud katsete arvu vastavalt tema vanusele.

3.1.2 Mittefunktsionaalsed nõuded

Tavakasutaja mittefunktsionaalsed nõuded:

- Tavakasutajana soovin veebirakendust kasutada vabalt valitud kaasaegses veebilehitsejas.

- Tavakasutajana soovin kasutada intuitiivset kasutajakogemust pakkuvat veebirakendust.

Administraatori mittefunktsionaalsed nõuded:

- Administraatorina soovin, et rakenduse lähtekood asuks kergesti hallatavas keskses koodihoidlas.
- Administraatorina soovin, et rakendusele oleks koostatud põhjalik dokumentatsioon, et saaksin tulevikus vähese vaevaga uusi inimesi arendusse kaasata või vajadusel oma positsioon üle anda.

3.1.3 Tulevikus loodavate funktsionaalsuste nõuded

Ajalise piirangu tõttu jäävad mõned planeeritavad funktsionaalsused käesoleva lõputöö skoobist välja, kuid on rakenduse potentsiaali tutvustamise huvides järgnevalt siiski välja toodud. Need hõlmavad registreeritud kasutaja rolli loomist, mis ei ole rakenduse baasfunktsionaalsuste toimimiseks vajalik, kuid loob võimalused veelgi mitmekesisemaks algoritmilise mõtlemise arendamiseks. Samuti kuuluvad siia kasutajate haldust puudutavad nõuded administraatori rolli jaoks.

Tavakasutaja funktsionaalsed nõuded:

- Tavakasutajana soovin end registreeritud kasutajaks registreerida.

Registreeritud kasutaja funktsionaalsed nõuded:

- Registreeritud kasutajana soovin sisse ja välja logida.
- Registreeritud kasutajana soovin mänge lisada.
- Registreeritud kasutajana soovin enda loodud mänge näha.
- Registreeritud kasutajana soovin enda loodud mänge muuta.
- Registreeritud kasutajana soovin enda loodud mänge kustutada.
- Registreeritud kasutajana soovin taotleda enda loodud mängu avalikustamist kõigile kasutajatele.
- Registreeritud kasutajana soovin näha teiste loodud mänge, mille administraator on kinnitanud ja avalikustanud.

Registreeritud kasutaja mittefunktsionaalsed nõuded:

- Registreeritud kasutajana soovin, et minu isikuandmed oleksid turvaliselt hoiustatud.

Administraatori funktsionaalsed nõuded:

- Administraatorina soovin registreeritud kasutaja loodud mängu avalikustamistaotluse kinnitada.
- Administraatorina soovin registreeritud kasutaja loodud mängu avalikustamistaotluse tagasi lükata.
- Administraatorina soovin registreeritud kasutajale anda laiendatud õigused.
- Administraatorina soovin registreeritud kasutajalt võtta laiendatud õigused.

3.2 Tehnoloogia valik

Kaasaegsete tehnoloogiate valik veebirakenduste loomiseks on lai ning kiiresti uuenev [11]. Vajaduse tõttu tulevikus tALGOtech rakendusse võimalus paindlikult uusi mänge lisada, on otstarbekas kasutajaliides ning äriloogika lahus hoida. Seetõttu arendatakse klient ja teenus eraldiseisvate rakendustena, mis lihtsustab kogu rakenduse hallatavust ning tulevikus funktsionaalsuste lisamist ja uute versioonide loomist. Käesoleva lõputöö raames on vaadeldud vaid kaasaegseid ning laia kasutajaskonnaga klientrakenduste loomiseks mõeldud tehnoloogiaid. Lühidalt kirjeldatakse ka serveripoolse rakenduse arenduseks tehtud tehnoloogilisi valikuid, mille täpsema analüüsi võib leida kaastudeng Maarja Helena Elisabeth Hoopi bakalaureusetööst.

3.2.1 Serveripoolse rakenduse tehnoloogiate valik

Serveripoolse rakenduse programmeerimiskeeleks valiti Java koos Spring Boot raamistikuga. Projekti arendatakse IntelliJ Idea keskkonnas ning koodi hallatakse Tallinna Tehnikaülikooli GitLabis. Valmis rakenduse andmebaasiks valiti MariaDB, kuid arendusperioodi jooksul otsustati selle paindlikumate muutmisvõimaluste tõttu kasutada mälupõhist HSQLDB-d.

3.2.2 Klientrakenduse raamistik

Kaks olulisemat programmeerimiskeelt veebilehtede arendamiseks on PHP ja JavaScript. Kuna aga esimene neist on loodud eelkõige veebirakenduste serveripoolseks arenduseks ning viimane klientrakenduste tegemise suunitlusega [12], kasutatakse tALGOtech klientrakenduse arenduseks JavaScripti. Koodi loetavuse ning rakenduse töökindluse parandamiseks seoses erinevate andmetüüpide manipuleerimisega [13] kirjutatakse projekti kood JavaScripti tüübitud edasiarenduses, TypeScriptis.

Klientrakenduse raamistiku valimiseks on populaarsuse järjekorras välja toodud 2021. aasta viis enimkasutatavat JavaScripti peale ülesehitatud klientrakenduste arenduseks mõeldud raamistikku [14] koos nende olulisemate omadustega:

- React.js – interaktiivsete kasutajaliideste loomise lihtsustamiseks mõeldud raamistik, mis on üles ehitatud eraldatud komponentide põhisele arhitektuurile, kus programmeerija ülesandeks on luua iga rakenduse oleku kohta vastav vaade. Eraldatud komponentidel põhinev ülesehitus hoiab rakenduse oleku veebilehitseja DOM-ist eraldatuna ja tagab nii klientrakenduse arhitektuuri kergesti hallatavuse kui ka mastaabitavuse. Seda tänu ainult uuenenud andmetega komponendi veebilehel uuesti laadimisele. Reacti on võimalik kasutada ka serveripoolse koodi ning mobiilirakenduste kirjutamiseks [15].

- jQuery – intuitiivsusele ning mitmekülgsele funktsionaalsusele rõhuv raamistik, mis lihtsustab API abil HTML dokumentide käitlemist, sündmuste töötlust ja AJAX tehnoloogia kasutamist [16].

- Angular – rakenduste mastaabitavuse üle head kontrolli andev raamistik [17], mis teeb kasutaja loodud mallid JavaScripti virtuaalmasinate jaoks optimeeritud koodiks. Lisaks sellele aitab jõudluse parandamisele kaasa ka ainult selle vaate laadimine, mida kasutaja parasjagu vajab [18].

- Vue.js – intuitiivsel API-l põhinev raamistik, mis tegeleb kasutaja eest kompilaatori jaoks koodi optimeerimisega. Kasuteguriks on ka võimalus raamistikku järkjärguliselt arendusprojekti kaasata, kuna nii saab seda vajadusel hõlpsasti koos teiste JavaScripti teekidega kasutada [19].

- ASP.NET Core – kergesti ka teiste JavaScriptile loodud teekidega ühilduv ning programmeerimiskeelte C# ning JavaScript kombineerimist võimaldav raamistik, millega saab luua nii kasutajaliideseid kui API-sid. Arendajate tööd lihtsustab raamistiku automatiseeritus kliendi- kui serveripoolsete andmete valideerimisel. Samuti hoolitseb antud teek HTTP päringust saabuvate andmete töötlemise eest rakenduses olevate meetodite parameetrite kujule [20].

Lõputöö ajalise piirangu tõttu on välja toodud raamistike seast valiku tegemisel oluline arvesse võtta ka autori kogemust nendega arendamisel ning igaühe õppimiskeerukust. Õppimiskeerukuse hindamisel on eeldatud, et õpilasel on olemas eelnev kokkupuude JavaScriptiga. Järgnevalt on välja toodud raamistike võrdlus tabelina (Tabel 1).

Tabel 1. Klientrakenduse raamistike võrdlus.

Raamistik	Kogemus	Õppimiskeerukus
React	Hea	Keskmine
jQuery	Puudub	Madal
Angular	Puudub	Keskmine
Vue.js	Rahuldav	Madal
ASP.NET Core	Rahuldav	Keskmine

Kõige põhjalikum on kokkupuude olnud React raamistikuga, veidi põgusam Vue.js ja ASP.NET Core teekidega. Angular on autorile võõras ning samuti ei ole tALGOtech rakenduse jaoks eriliseks probleemiks selle mastaabitavus, mis oleks üks põhilisi Angulari valimise kasutegureid. Samuti jääb valikust välja jQuery, mille kasutamisega kogemus puudub ning mille rakendused on tänapäeval võrreldes teistes raamistikes arendatutega pigem aeglased ja raskesti hallatavad ning mille tehnoloogiat loetakse pigem aeguvaks.

Võrreldes valikusse jäänud React, Vue.js ja ASP.NET Core raamistikke, on oluline arvesse võtta ka tALGOtech serverrakenduses kasutatavaid tehnoloogiaid. Kuna ASP.NET Core on loodud eelkõige eesmärgiga teha klientrakendusi C# programmeerimiskeeles kirjutatud teenusepoolsetele süsteemidele, kuid tALGOtech serverrakendus pole selles arendatud, jäetakse ASP.NET Core valikust välja. Vue.js ning

React raamistike seast eelistatakse viimast, kuna sellega on autoril laialdasem kogemus ning selle populaarsus on oluliselt kõrgem [21].

3.2.3 Klientrakenduse disainiraamistik

Eesrakenduste loomisel on oluline osa lõpptulemuse visuaalsel väljanägemisel. Ühtlast stiili aitab tekitada juba olemasolevaid kujunduskomponente pakkuvate disainiraamistike kasutamine. Ühtlasi optimeerivad need veebilehe esteetilise välimuse saavutamiseks vajamineva koodi hulka, muutes seeläbi kogu rakenduse kergemini hallatavaks. Valikusse on võetud 2022. aastaks kolmeks populaarsemaks prognoositud disainiraamistikud, mis on kõik loodud kasutamiseks koos klientrakenduse arenduseks valitud raamistiku Reactiga [22].

- MUI (*Material User Interface*) – teek, mis pakub arendaja jaoks valmis tehtud temaatikaga malle ning komponente. Lisandväärtuseks on asjaolu, et MUI on loodud Google Material Designi põhjal, tänu millele saab sellega soovi korral luua ka täiesti isikliku disainisüsteemi [23].

- React-Bootstrap – populaarne Bootstrap disainiraamistik, mis on täielikult ümber kirjutatud React raamistikku. Tänu sellele, et tegu on Bootstrapil põhineva raamistikuga, mille arendus on kestnud aastaid, võimaldab see kasutada tuhandeid Bootstrapile loodud teemasid ja komponente [24].

- Ant Design – professionaalse tasemega kasutajaliideste loomiseks mõeldud raamistik, mis pakub kujunduste loomiseks kõrgetasemelisi Reacti komponente, mida on võimalik soovi korral kohandada [25].

Kõik disainiraamistikud pakuvad laia valikut teemasid ning komponente, mida on mugav React raamistikuga arendatava klientrakenduse kujunduse loomisel kasutada. Valiku tegemise kriteeriumiteks võetakse autori kogemus nende kasutamisel ja iga disainiraamistiku baasseadistustega maht, säilitamaks rakenduse optimaalset jõudlust. Samuti arvestatakse 5-pallisel skaalal subjektiivset sümpaatiat raamistike pakutavate komponentide väljanägemise suhtes, kus “5” on kõrgeim võimalik hinnang. Järgnevalt võrreldakse kõiki raamistikke nimetatud kriteeriumite alusel tabelina (Tabel 2).

Tabel 2. Klientrakenduse disainiraamistike võrdlus.

Raamistik	Kogemus	Komponentide välimus
MUI	Puudub	5
React Bootstrap	Puudub	2
Ant Design	Hea	4

Ehkki autoril on kogemus vaid Ant Design raamistiku kasutamisel, ei välista see teiste raamistike valimist, kuna kõigil on olemas põhjalik dokumentatsioon, mille alusel on kerge ka teisi kasutama õppida. React Bootstrap raamistiku välistab selle pakutavate komponentide baasvälimus. Võrreldes Ant Design ning MUI teeke on autoril esimesega küll rohkem kogemust, kuid kuna viimase komponentide välimus on autorile sümpaatsem ning paistab lapselikum, valitakse klientrakenduse mängufunktsiooni silmas pidades just MUI.

3.2.4 Klientrakenduse animeerimisraamistik

tALGOtech eesrakenduse oluliseks funktsionaalsuseks on kasutajale tema loodud programmi toimimise illustreerimine kergesti mõistetava animatsiooniga. Selle jaoks on vajalik leida sobiv veebilehtedele graafika loomiseks ning animeerimiseks mõeldud raamistik. Järgnevalt on välja toodud 2020. aasta populaarsematest Javascripti graafikaraamistikest kaheksa, mis on mõeldud joonistuste ning animatsioonide loomiseks [26]:

- Three.js – teek, mis on mõeldud kolmedimensioonilise graafika animeerimiseks ning veebilehtedele paigutamiseks [27].
- GSAP (*The GreenSock Animation Platform*) – nii DOM elementide, teiste graafikaraamistikega loodud kujutiste kui Javascripti objektide animeerimist võimaldav raamistik, millega saab animatsioonis osalevatele objektidele realistlikkuse huvides ka füüsikalisi omadusi lisada [28].
- Two.js – kahedimensiooniliste joonistuste loomiseks ning animeerimiseks mõeldud raamistik [29].

- Pts.js – kahedimensiooniliste joonistuste loomise ning animeerimise raamistik, mis kasutab joonistuste ning loodud liikumiste ühildamiseks ainulaadset ruumist, vormist ning punktist lähtuvat metoodikat [30].
- Anime.js – animatsioonipakett, mis võimaldab lihtsa API-ga kontrollida CSS omadusi, SVG-sid, DOM-atribuute ning JavaScripti objekte [31].
- PixiJS – API-põhine graafikaraamistik, mis pakub mitmekesist ning kõrgetasemelist pilditöötluse võimalust [32].
- Zdog – näiliselt kolmedimensiooniliste kujutiste loomiseks ning animeerimiseks mõeldud raamistik, mis on tänu tegelikule kahedimensioonilisusele siiski kiire ega langeta rakenduse jõudlust [33].
- Snap.svg – SVG objektide animeerimiseks mõeldud teek [34].

Oluline on ühilduvus klientrakenduse kirjutamiseks valitud raamistiku – Reactiga, mis on olemas kõigil loetletud pakettidel. Kuna töö autoril pole ühegi animeerimisraamistikuga varasemat kokkupuudet, on valiku tegemisel arvesse võetud peamiselt iga paketi pakutava funktsionaalsuse vastavust rakenduse tegelikele vajadustele ning erinevate pakettide õppimiskeerukust.

Järgnevalt võrreldakse kõiki raamistikke mainitud tegurite alusel tabelina (Tabel 3). Hinnangu "liiga kitsas funktsionaalsus" andmise põhjuseks on juba olemasoleva, näiteks veebist alla laetud kujutise manipuleerimise võimaluse puudumine. See on rakenduse edasiarenduse võimalusi silmas pidades oluline tegur, sest muidu pole võimalik tulevikus kasutajate loodud uutele mängudele dünaamiliselt animatsioone luua.

Tabel 3. Klientrakenduse animeerimisraamistike võrdlus.

Raamistik	Funktsionaalsuse vastavus vajadusele	Õppimiskeerukus
Three.js	Liiga lai funktsionaalsus	Kõrge
GSAP	Sobiv	Madal
Two.js	Liiga kitsas funktsionaalsus	Keskmine

Pts.js	Sobiv	Kõrge
Anime.js	Sobiv	Madal
PixiJS	Ebasobiva suunitlusega funktsionaalsus	Keskmine
Zdog	Liiga kitsas funktsionaalsus	Keskmine
Snap.svg	Sobiv	Madal

Jättes välja kõik ebasobiva funktsionaalsuse ja kõrge õppimiskeerukusega raamistikud, jäävad valikusse GSAP, Anime.js ning Snap.svg. Viimase kahjuks räägib selle kasutatavate tehnoloogiate vananemine ning võimaluse puudumine manipuleerida muid objekte peale SVG. Seetõttu langeb ka see valikust välja. Võrreldes GSAP-i, mille mõningad funktsionaalsused on kättesaadavad vaid tasu eest, ning täielikult vabavaralist Anime.js-i, eelistatakse viimast. Paljude animeerimisraamistike mahukuse tõttu tehti projekti optimaalse suuruse huvides kindlaks, et valitud raamistik poleks liialt suur [35].

3.3 Arenduskeskkonna valik

Arenduskeskkonna valikul analüüsitakse eraldi enimkasutatavaid koodihalduseks mõeldud vahendeid ning integreeritud arenduskeskkondi. Otsuse tegemisel võetakse lisaks muule arvesse ka tööriistale tasuta ligipääsu võimalusi.

3.3.1 Koodihalduse keskkonna valik

Võrreldud on kolme populaarset koodihalduse tööriista [36], millest kõigiga on käesoleva lõputöö autoril olemas ka isiklik kasutuskogemus.

- GitHub – üks suurimaid ja arenenumaid koodihalduskeskkondi, mida kasutab üle 73 miljoni arendaja ning 4 miljoni ettevõtte. See pakub mitmekülgseid lahendusi projekti arendamise käigus vajalikuks suhtlemiseks, võimaldades koodi mugavat ning funktsionaalset tagasisidestamist. Lisandväärtusena integreerub see väga paljude teiste keskkondadega [37].

- Bitbucket – 10 miljoni kasutajaga koodihalduskeskkond [38] on samuti loodud eesmärgiga aidata meeskondadel ühise arendusprotsessi töövoogu hallata. Ainulaadsete eelistena võib välja tuua koodi kvaliteeti hindavate raportite genereerimise võimaluse, arendajate õiguste haldamise süsteemi ning hea integreerituse populaarse projektijuhtimist lihtsustava tööriista Jiraga [39].

- GitLab – ligikaudu 30 miljoni kasutajaga avatud koodibaasiga kiiresti edasi arenev koodihalduskeskkond [40]. See pakub lisaks projekti arendusaegsele haldusele ka planeerimistööriistu ning võimalusi valminud projekti statistika jälgimiseks [41].

Valikus olevatest koodihalduse keskkondadest on eelistatuim GitLab, kuna see pakub parimaid võimalusi kogu projekti elutsükli haldamiseks ning on erinevalt teisest kahest nimetatud keskkonnast üksikisikutele täies mahus tasuta kättesaadav. Täpsemalt hallatakse projekti koodi Tallinna Tehnikaülikooli serveri GitLabis, kuna see võimaldab ligipääsu vaid töötajatele ja tudengitele ning potentsiaalsete edasiarendajatena nähakse just tulevasi üliõpilasi.

3.3.2 Integreeritud arenduskeskkonna valik

Integreeritud arenduskeskkondadest on vaatluse alla võetud Visual Studio Code ja WebStorm, mis on ühed arenenumad JavaScriptis kirjutatud klientrakenduste tegemiseks mõeldud keskkonnad ning mõlemad ka põhjalikult dokumenteeritud [42].

- WebStorm – arenduskeskkond, mis on ehitatud avatud lähtekoodiga IntelliJ platvormile ja pakub erinevaid arendust lihtsustavaid funktsionaalsusi nagu vigade tuvastamine, koodi kvaliteedi analüüs ning kontrollitud refaktoreerimine. Samuti on loodud võimalus mugavaks samm-sammuliseks vigade tuvastamiseks ja keskkonda sisseehitatud HTTP kliendiga rakenduse päringute testimiseks [43].

- Visual Studio Code – Microsofti loodud keskkond, mis kasutab arendusprotsessi hõlbustamiseks IntelliSense tarkvara. See sisaldab endas koodi semantilist analüüsi, kirjutatu automaatjätkamist ning võimalusi refaktoreerimiseks. Eeliseks on ka keskkonna kiirus ning integreerimisvõimalus paljude teiste tehnoloogiatega [44].

Käesoleva lõputöö klientrakenduse integreeritud arenduskeskkonnaks on valitud Visual Studio Code, millega on autoril olemas põhjalik kasutuskogemus ning mis on erinevalt WebStormist tasuta.

3.4 Veebirakenduse disain

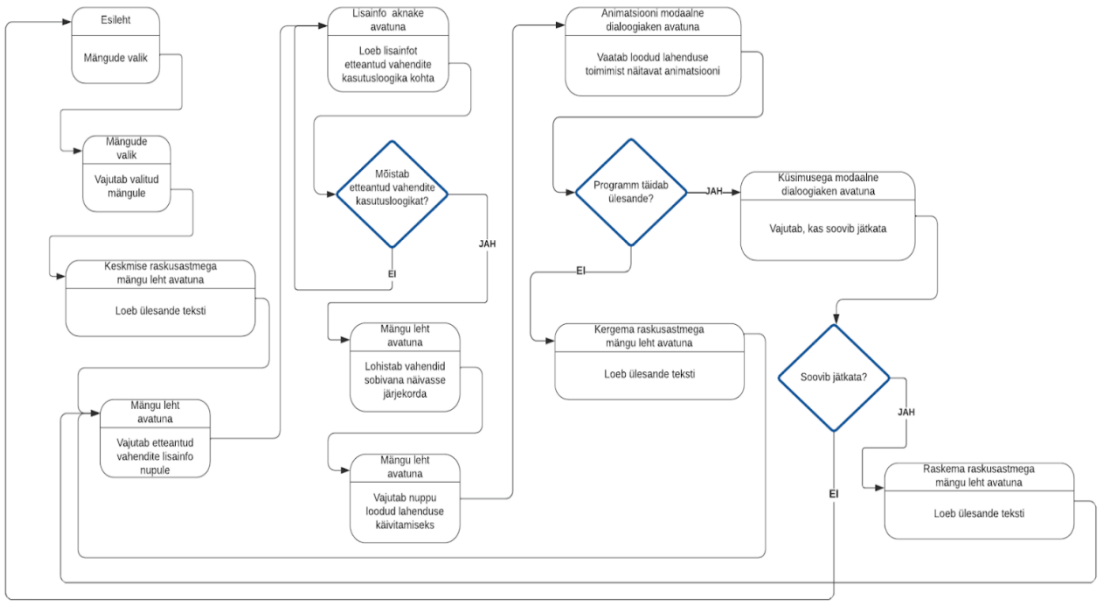
Antud veebirakendus on mõeldud kasutamiseks eeskätt ajaliselt piiratud töötubade raames. See eeldab vähest ajakulu rakenduse funktsionaalsustega tutvumiseks, et esmakordsel kasutajal oleks võimalik kohe ülesande lahendamisele keskenduma hakata. Veebilehe intuitiivsuse ning kergesti hoomatavuse saavutamiseks on läbi viidud kolm sammu:

- Kasutajakogemuse disain.
- Kasutajakogemuse visualiseerimine Figma tarkvaraga.
- Andmebaasi mudeli loomine.

3.4.1 Kasutajakogemuse disain

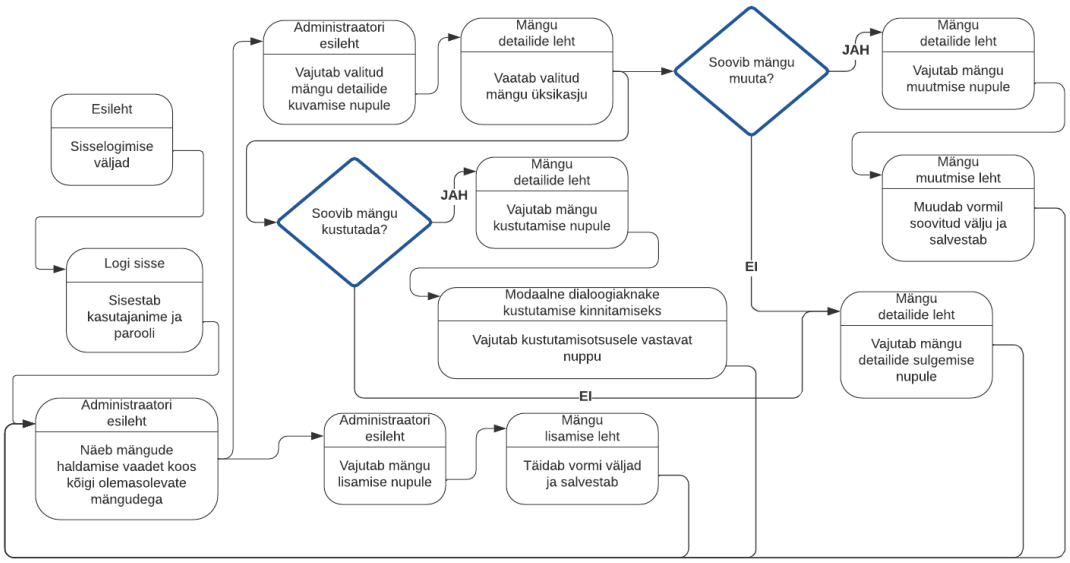
Kasutajalugudest lähtuvalt jaotuvad käesoleva lõputöö skoopi jäävad vaated kaheks: üks tavakasutajale ning teine administraatorile. Tavakasutaja vaade hõlmab valitud ülesande lahendamist, administraatori oma sisse- ja väljalogimist ning mängude haldust.

Joonisel 1 on välja toodud kasutajakogemus, mis keskendub tavakasutaja ülesande lahendamise voole.



Joonis 1. Tavakasutaja ülesande lahendamise kasutajakogemuse diagramm.

Joonisel 2 on kujutatud administraatori kasutajakogemust, mis võimaldab lisaks sisselogimisele ka mängude vaatamist, lisamist, muutmist ning kustutamist.



Joonis 2. Administraatori kasutajakogemuse diagramm.

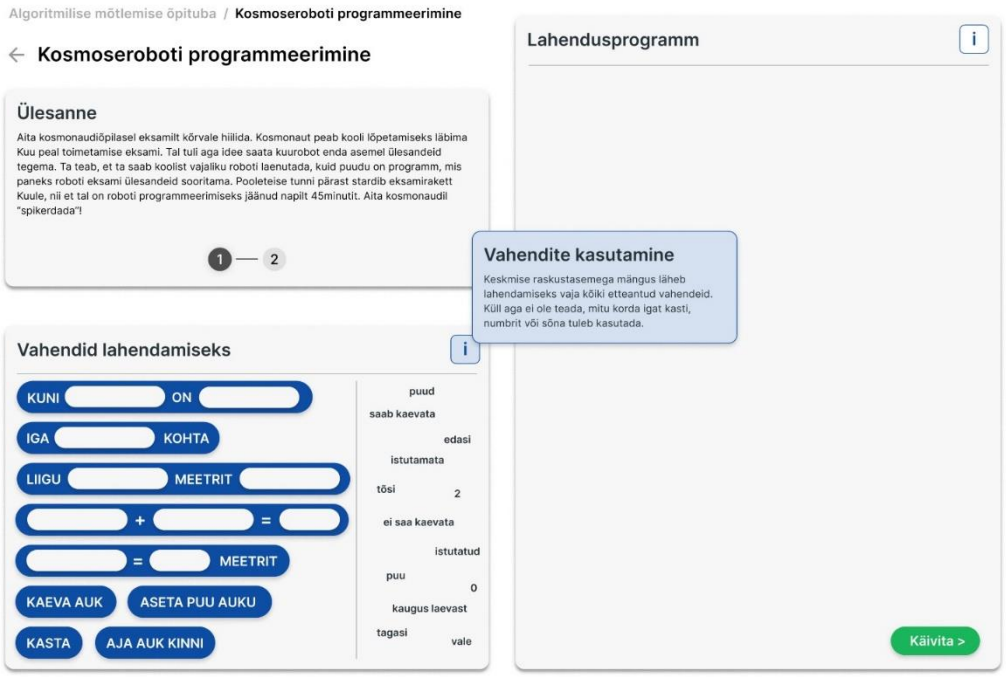
3.4.2 Kasutajakogemuse disain Figma tarkvara abil

Figma on populaarne veebipõhine tööriist kasutajaliideste disainimiseks ning interaktiivsete prototüüpide loomiseks, mis võimaldab ka mitme inimese samaaegset töötamist ühe projekti kallal [45].

Kasutajalugudest lähtuva prototüübi loomine on oluline tegur rakenduse esialgse visiooni testimiseks, kuna selle põhjal on kerge tuvastada vajalikke muutusi. Kui avastada samad kitsaskohad arenduse käigus, on nende ümber tegemine tunduvalt keerulisem ning ressursikulukam. Samuti aitab prototüüp tunduvalt lihtsustada klientrakenduse arendust, kuna annab hea ettekujutuse loodava rakenduse komponentidest ning nende toimimiseks vajalikest andmetest.

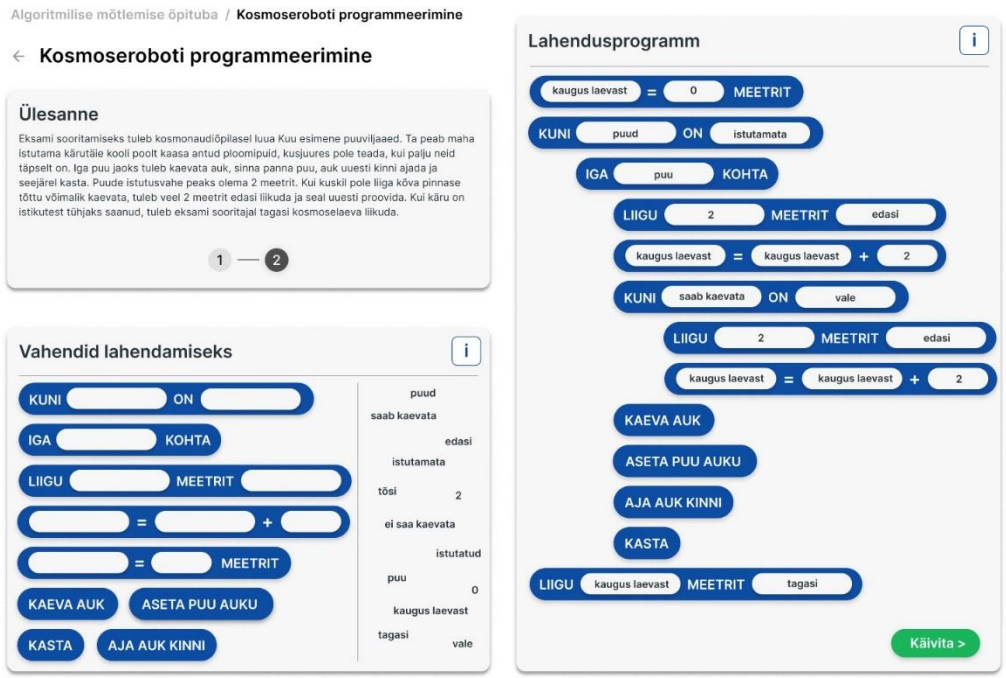
Käesolev prototüüp on loodud, kasutades arendatava rakenduse tehnoloogiavalikusse kuuluva disainiraamistiku MUI komponente, et lõpptulemusest võimalikult hea ettekujutus saada.

Joonis 3 kujutab vaadet, mis avaneb kasutajale pärast kindla mängu valimist. Ülesande lahendamist pole veel alustatud, kuid kasutaja on täpsustavate juhiste saamiseks avanud lahendusvahendite juures oleva infoaknakese.



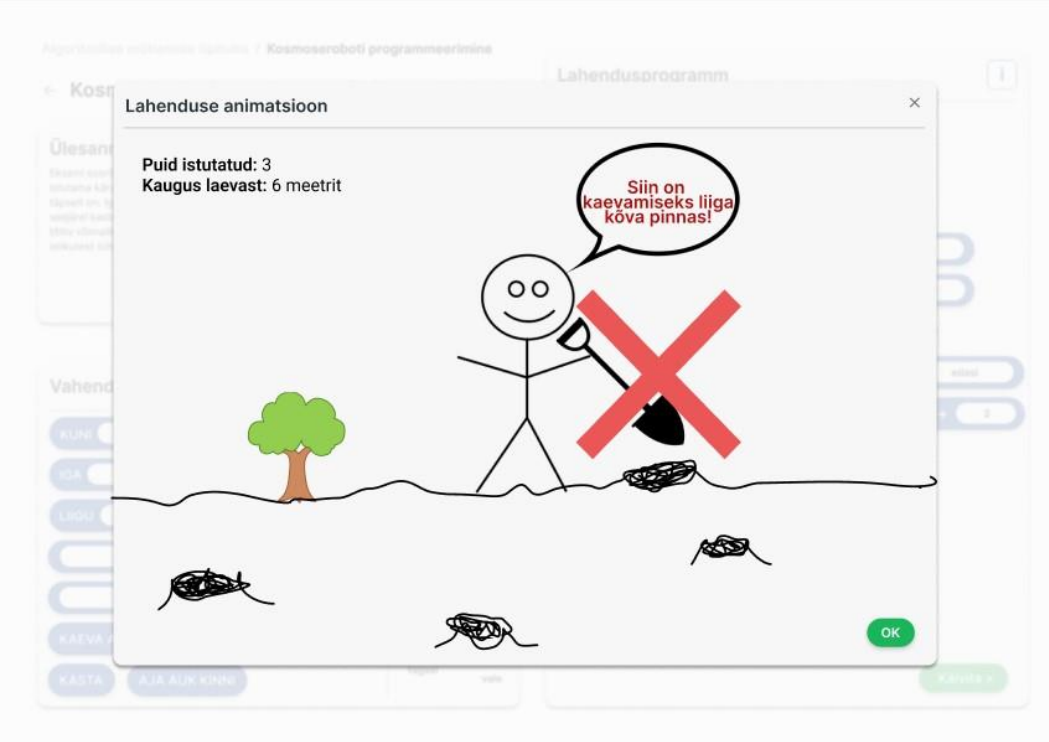
Joonis 3. Tavakasutaja ülesande lahendamise vaate prototüüp enne lahendamist.

Joonis 4 näitab ülesande vaadet koos kasutaja koostatud lahendusega.



Joonis 4. Tavakasutaja ülesande lahendamise vaate prototüüp koos lahendusega.

Joonisel 5 on näidatud vaade, mida kasutaja näeb oma programmi käivitades loodud algoritmi tagasisidena. Kujutatud olukorras on kasutaja algoritmi põhjal istutatud kolm puud, kuid kõva pinnase tõttu on edasine istutamine võimatu. Kasutaja pole algoritmi sisse kirjutanud enne kaevama asumist pinnase seisukorra kontrollimist ning vajadusel uue koha otsimist, mistõttu pole ülesanne sooritatud.

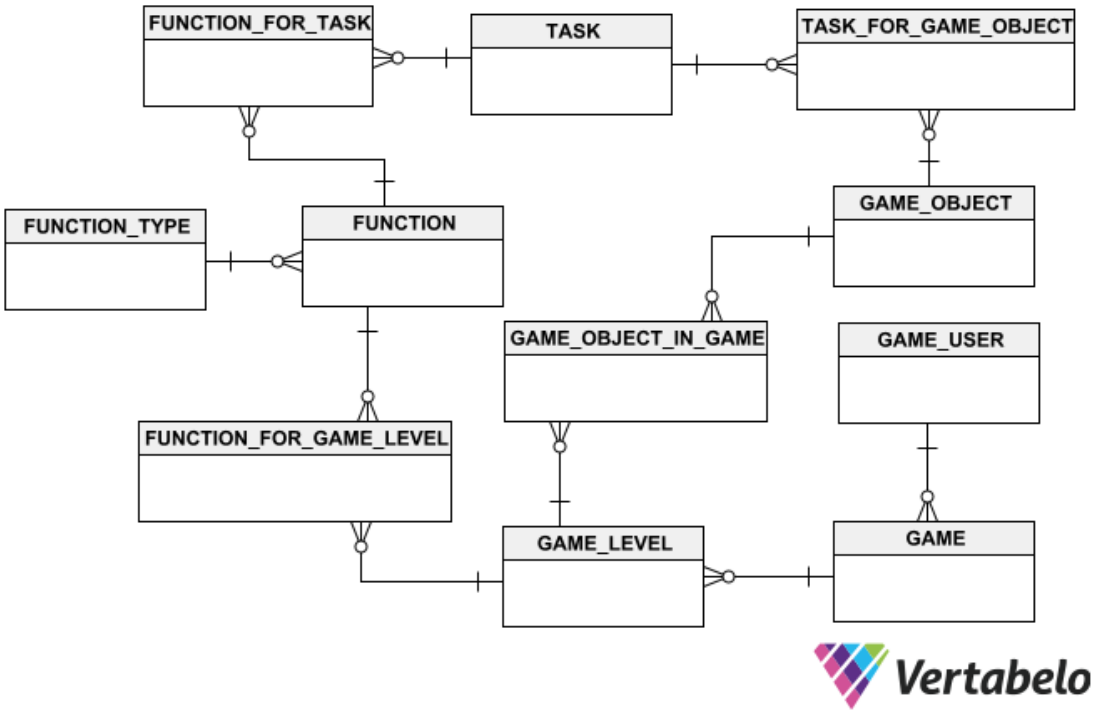


Joonis 5. Tavakasutaja vaate prototüüp loodud algoritmi animatsiooni vaatamiseks.

3.4.3 Andmebaasi mudel

Pidades silmas kasutajalugusid ning kasutajakogemuse disaini, loodi vajalikke olemeid sisaldav sobiva struktuuriga andmebaasi mudel. Olemi-suhte diagramm on koostatud Vertabelo tööriistaga.

Joonisel 6 on kujutatud lihtsustatud tALGOtech rakenduse andmebaasi skeem. Täielik skeem on lisas 3.



Joonis 6. Lihtsustatud olemi-suhte diagramm.

Andmebaasi olemite täpsema funktsionaalsuse kirjelduse ning andmebaasi valiku analüüsi võib leida lõputöö kaasautori Maarja Helena Elisabeth Hoopi tööst.

3.5 Analüüsi kokkuvõte

Analüüsi käigus tehti kindlaks rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded erinevate kasutajagruppide jaoks, millest lähtuvalt langetati lõputöö raames valmiva klientrakenduse tehnoloogilisi valikuid ning loodi eeldatav kasutajakogemuse disain ja kasutajaliidese prototüüp.

Programmeerimiskeeltest osutus valituks JavaScriptile üles ehitatud TypeScript koos React raamistikuga selle korduvkasutatavate komponentide põhise ülesehituse tõttu, tänu millele on rakendus kergesti hallatav ja mastaabitav. Samuti on Reacti suureks eeliseks selle populaarsus ja autori kogemus.

Kasutatavaks disainiraamistikuks valiti MUI, peamiselt selle küllaltki väikese mahu ning subjektiivse sümpaatia tõttu pakutavate elementide välimuse suhtes. Lisaks võimaldab see tulevikus rakendusele soovi korral unikaalse disainisüsteemi luua.

Sisendi alusel loodud animatsiooni teostamiseks otsustati kasutada Anime.js teeki, kuna see on täielikult vabavaraline ning võimaldab manipuleerida paljusid erinevaid objekte. Samuti põhineb Anime.js küllaltki intuitiivsel API-l ning madal õppimiskeerukus on autori kogemuste puudumise ja lõputöö ajalise piiratuse tõttu tähtis.

Koodi kirjutatakse Visual Studio Code arenduskeskkonnas ning hallatakse Tallinna Tehnikaülikooli GitLabis.

4 Klientrakenduse arendus

Klientrakenduse loomisel peeti silmas analüüsi käigus välja selgitatud nõudeid ning tehniliste vahendite valikut. See on arendatud veebiteenusest täielikult eraldiseisvalt ning kasutab serveripoolse rakendusega suhtlemiseks kindlaid ühenduspunkte, mille kaudu vahetatavad andmekujud vastavad nii klient- kui ka serverirakenduse vajadustele.

Veebileht on ülesehitatud eraldiseisvaid komponente kombineerides, mida on kõiki võimalik taaskasutada. See on valitud arendusraamistiku üks peamisi eeliseid: nii on rakendus kergemini hallatav ning optimaalsema jõudlusega. Viimase tingib Reacti olekupõhine elutsükkel. See tähendab, et komponentides on võimalik määrata muutujad, mille seisu jälgitakse ning nende uuenemisel ei laeta uuesti mitte kogu lehte, vaid ainult uut väärtust sisaldav alamkomponent.

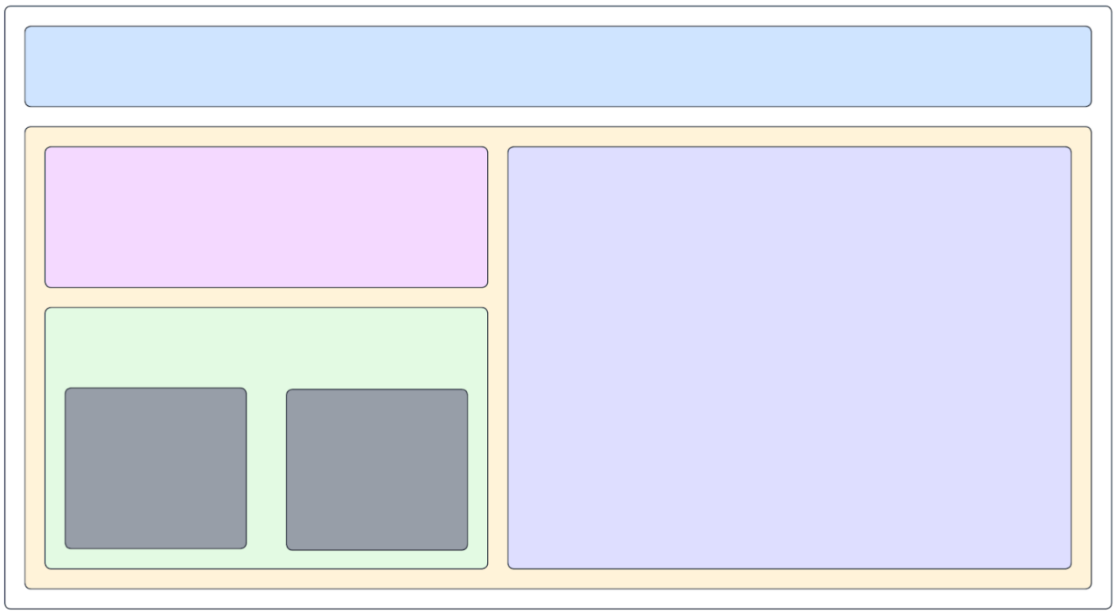
Käesoleva lõputöö projekt on loodud Create React App lahenduse abil, mis võimaldab klientrakenduse baasprojekti luua mugavalt ja kiirelt käsurea kaudu, nõudmata liigset seadistamist. Kuna lõputöö arendati Typescriptis, mitte JavaScriptis, lisati projekti loomise käsule soovitud tulemuse saamiseks vastav parameeter. Käsu käivitamiseks ning edasiseks arenduseks on arvutis vajalik npm-i ehk *Node Package Manager*i olemasolu, mis haldab projektis olevaid teeke ning käitab serverit rakenduse arendamiseks [46].

Kuna klientrakenduse arendus toimus paralleelselt serveripoolse teenuse loomisega, kasutati arenduse käigus fiktiiv-tagarakendust. Selleks tehti eraldi projekt JSON Server lahenduse abil, mis on mõeldud serveripoolsete rakenduste matkimiseks, tagastades ettenähtud päringute saabudes sinna kirja pandud andmeid. Fiktiiv-tagarakenduse tegemisel lähtuti reaalse tagarakenduse arendajaga eelnevalt kokkulepitud ühenduspunktidest ning andmekujudest [47].

4.1 Komponentide planeerimine

Komponentide hierarhia planeerimist ühe esimese sammuna peetakse React rakenduse loomisel heaks praktikaks. Arendatava lehe komponentideks jaotamisel tuleb tähele panna ühe vastutusala printsiipi ehk lahutada kasutajaliides komponentideks, kus igaüks neist tegeleks ühe andmemudeli osa näitamisega [48].

Joonisel 7 on kujutatud alamleht, mis kuvatakse kasutajale valitud mängu ülesande lahendamiseks. Valge kast hõlmab kogu lehte, mis peaks tulevikus hoiustama teavet kasutajate autentimisinfo kohta, kuid on lõputöö skoobis valminud rakenduses olekuta. Helesinine kast on lehekülje päis, mis on samuti olekuta, saades navigeerimiseks vajaliku näidatava informatsiooni lehekülje aadressiribalt. Kollane kast on spetsiifiline valitud mängu vaatele. Selle olek sisaldab andmebaasist päritud mängu ning sessiooni mälust saadud lahenduse informatsiooni, millest vajalikke osi omakorda erinevatele alamkomponentidele vahendatakse. Roosale kastile antakse olekusse kaasa mängu ülesande tekst. Rohelisse komponenti liiguvad andmed mängu ülesande lahendamiseks ettenähtud vahendite ja nende kasutamise kohta, kusjuures vasakpoolne hall alamkomponent võtab vastu lahenduskastide ning parempoolne lüngatäidete loendi. Lisaks sellele edastatakse kumbagi halli kasti teave kasutaja loodud lahenduse kohta, mis on vajalik kergel tasemel juba kasutatud vahendite desaktiveerimiseks. Lilla kast on mõeldud lahenduse tegemiseks ning saab kollaselt komponendilt informatsiooni käimasoleva mängu identifikaatori ja eelnevalt loodud lahenduse kohta, et näiteks lehe värskendamisel kasutaja loodud lahendus säiliks.



Joonis 7. Valitud mängu ülesande lahendamise alamlehe React komponentide paiknemine.

4.2 React – *Single Page Application (SPA)*

SPA ehk *Single Page Application* on klientrakenduse ülesehituse üks tüüpe, mis on tänapäeval väga laialt levinud. Sel viisil tehtud veebileht laeb korraga ära kogu HTML ning Javascript koodi ning kasutaja lehel navigeerides muudab näidatavat sisu vajadusele vastavalt ilma, et peaks selleks lehte värskendama [49].

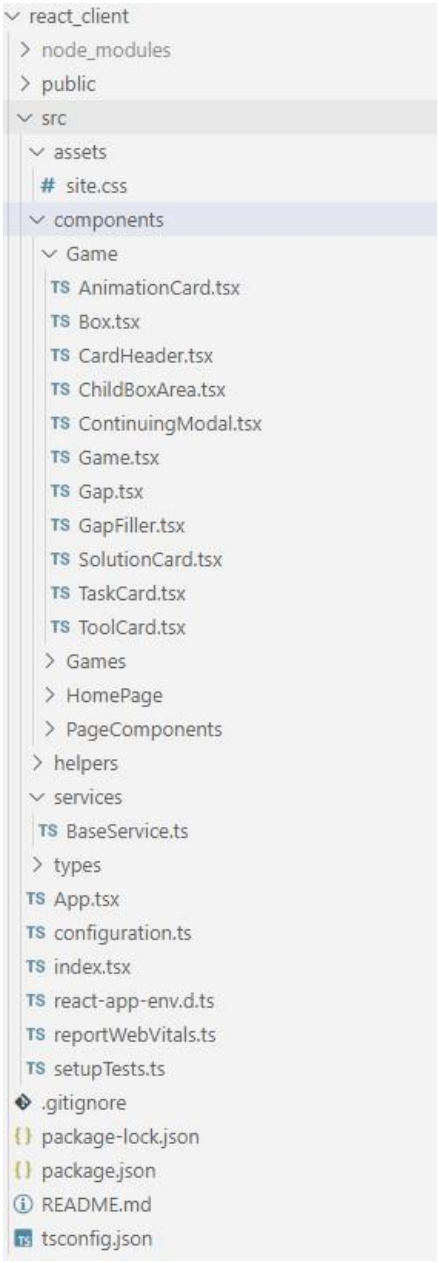
Selle alternatiiv on MPA ehk *Multi Page Application*, mis iga uue sisu näitamise vajaduse korral veebilehte värskendab. See vähendab oluliselt kasutajakogemuse sujuvust, kuna tekitab lehel navigeerimise korral iga kord kiire, kuid nähtava vilgatuse [49].

Lõputöö klientrakendus loodi SPA põhimõttel selle tunduvalt parema kasutajamugavuse tõttu. Ehkki rakendusel tuleb sooritada päringuid andmebaasile, millest saabuva sisu kuvamiseks võib aega minna, on võimalik kasutajale näidata andmetest sõltumatuid lehe komponente ning vajalikus kohas päringuprotsessi kajastavat laadimisrõngast.

4.3 Rakenduse struktuur

Create React App abil loodud baasprojektis on edasiseks arenduseks muuhulgas olemas lehekülje mall *index.html* genereeritavate html-elementide näitamiseks ja *index.tsx*, mis võimaldab TypeScripti kasutust [50]. React rakendused paiknevad ühes DOM juurelemendis, mille siseselt ei looda uusi veebilehitseja DOM elemente, vaid Reacti omi, millele vastavalt React Virtual DOM veebilehe lõplikku sisu kohandab. Tänu Reacti elementide objekti-olemusele on selline käitumine jõudluse poolest tunduvalt optimaalsem [51].

Rakenduse arenduse käigus lisanduvad vajalikud komponendid, tüüpe määravad liidesed ning abiklassid projekti „src“ kausta. Käesoleva klientrakenduse struktuuris, mis on kujutatud joonisel 8, on komponendid alamkaustadesse jaotatud vastavalt neid kuvavatele alamlehtedele. Eraldi on kokku koondatud rakenduseüleselt vajaminevad abimeetodid, andmetüübid ning päringute tegemise loogika.



Joonis 8. tALGOtech klientrakenduse projekti struktuur.

4.4 MUI disainiraamistiku kasutus

MUI on Google Material Designi põhjal loodud teek, mis pakub arendaja jaoks valmis tehtud temaatikaga malle ning erineva funktsionaalsusega [23]. Selle kasutamiseks tuleb käsureal projekti kaustas käivitada vastav käsk, mille tulemusena lisatakse teek sõltuvusena package.json faili. See hoiab endas metaandmeid, mida on vaja rakenduse *Node Package Manager*'ile avalikustamiseks ning määrab muuhulgas ära projekti toimimiseks vajalikud sõltuvused ja soovitud käitumise erinevate käskude käitamise korral [52].

Teatud komponentide välimuse mõningaseks muutmiseks, peamiselt suuruse ja värvi osas, on projektis CSS fail site.css, kus on täpsustatud vajalike komponentide stiiliparameetrid. Mõningate näitajate kohandamiseks, mille vaikimisi käitumine on MUI raamistiku poolt rangemalt määratud, tuleb soovitud väärtused täpsustada otse komponendi juures *style* karakteristiku kaudu.

4.5 Suhtlus veebiteenusega

Server- ja klientrakenduse vaheline suhtlus toimub läbi kindlate ühenduspunktide, mille kaudu saadetavad andmekujud on määratud vastavalt nende toel toimiva funktsionaalsuse vajadustele. Andmete vahetamiseks on loodud kõiki vajalikke päringuid koondav abiklass, mille meetodeid projekti üleselt soovitud parameetritega kasutatakse.

4.5.1 Päringute tegemine klientrakenduses

Klientrakenduses kasutatakse päringute tegemiseks Axios HTTP klienti, mis töötab rakenduses node.js-i HTTP mooduli abil ja veebilehitsejas XMLHttpRequest objektide kaudu. Axios põhineb Javascripti Promise objektidel, tänu millele on mugav tehtud päringu õnnestumist või nurjumist hallata [53].

Päringute mugavamaks haldamiseks ning paindlikumaks kasutamiseks on rakendusse loodud baasklass, mille meetodid määravad, kuidas vajalikke päringuid teha ning nende tulemusi töödelda. Kõik päringud võtavad sisendiks soovitud aadressi ning samuti tüübiparameetri, tänu millele on võimalik neid mugavalt kasutada projekti erinevates kohtades mistahes andmekujude saatmiseks ning pärimiseks. Samuti on päringutele sisendparameetrite ning vajaliku abimeetodi näol lisatud võimalus kasutada tulevikus turvalisuse tagamiseks JWT võtmeid. Kuna autenditud kasutajate süsteem jäi aga lõputöö skoobist välja, pole see hetkel projektis kasutusel. Joonisel 9 on kasutaja loodud lahenduse saatmiseks mõeldud päring, mis tehakse animatsiooni komponendis, näitamaks kasutajale tema tehtud algoritmi toimimist.

```
const result = await BaseService.post<IBox[],
ISolutionResult>('/games/' + props.gamePlayId, props.solution);
```

Joonis 9. Kasutaja loodud lahenduse saatmise päring animatsiooni komponendis.

Käesolevas rakenduses on oluline, et POST meetodiga tehtud päring võimaldaks saata üht ning tagastada teist tüüpi andmeid, mistõttu võtab joonisel 10 väljatoodud reaalselt tehtav päring vastu kaks tüübiparameetrit.

```
static async post<TEntity, TResult>(apiEndpoint: string, entity:
TEntity, jwt?: string): Promise<IFetchResponse<TResult>> {
	try {
		const response = await this.axios.post(apiEndpoint,
entity,			BaseService.getAxiosConfiguration(jwt));
		return {
			ok: response.status <= 299,
			statusCode: response.status,
			data: response.data,
			message: response.statusText,
		};
	} catch (err) {
		let error = err as AxiosError;
		return {
			ok: false,
			statusCode: error.response?.status ?? 500,
			message: error.response?.statusText
		};
	}
}
```

Joonis 10. POST päring päringute tegemiseks loodud baasklassis.

4.5.2 Ühenduspunktid ja vahetatavad andmekujud

Lõputöö raames valminud klientrakendus suhtleb serveripoolse teenusega nelja ühenduspunkti kaudu.

Esimene neist on aadressile '/*games'* tehtav GET päring, mille käigus päritakse kõikide rakenduses olevate mängude põhiline info. Saadud andmete töötlemiseks on loodud IGameInfo-nimeline liides, mida võib näha joonisel 11. Seda kasutatakse olemasolevate mängude nimekirja näitamiseks.

```
export interface IGameInfo {
	id: string;
	name: string | null;
	description: string | null;
	author: string | null;
	errorMessage: string | null;
}
```

Joonis 11. Iga mängu põhilise info liides.

Mängu valimisel kontrollitakse enne päringu tegemist, kas lahtioleval vahelehel sisenetakse mängu esmakordselt või on kasutajal juba mängimine pooleli. Esmasel laadimisel tehakse vastava mängu id-parameetriga GET päring aadressile '/*games*/{*id*}'. Vastuses sisalduvate andmete liides IGame on välja toodud joonisel 12.

```
export interface  IGame {
	id:  string;
	name: string | null;
	gamePlayId: string | null;
	errorMessage: string | null;
	author: string | null;
	description: string | null;
	currentLevel: {
		id: string;
		name: string;
		attemptsLeft: number;
		rules: string;
		task: string;
	} | null;
	boxes: IBox[] | null;
	gapFillers: IGapFiller[] | null;
}
```

Joonis 12. Valitud mängu kogu info liides.

Juhul, kui kasutaja on mängu juba eelnevalt mänginud, tehakse päring tema poolelioleva mängu taastamiseks. Selleks kasutab rakendus brauseri sessiooni mälu: mängu avades kontrollitakse, kas sessiooni mälus on valitud mängu id-parameetri võtmele vastavat väärtust. Kui vastav väärtus leitakse, tehakse eelnevalt katkestatud mängu jätkamiseks GET päring aadressile '/*gamePlay*/{*gamePlayId*}'. Väärtuse puudumise korral on tegu mängu esmakordse laadimisega ning tagarakendusest saadud gamePlayId, mis on alustatud mängukorrale unikaalne, salvestatakse edaspidiseks sessiooni mällu. Sel viisil on võimalik vältida alustatud mängukorra kadumist mängust lahkumisel või lehe värskendamisel. Kirjeldatud päringute tegemine on näha joonisel 13.

```
if (getGamePlayIdFromSessionStorage(id) !== null) {
	let gamePlayId = getGamePlayIdFromSessionStorage(id)!;
	result = await BaseService.get<IGame>('/gamePlay/' +
gamePlayId);
} else {
	result = await BaseService.get<IGame>('/gamePlay/' + id);
}
```

Joonis 13. Vajaliku päringu tuvastamine ja teostamine mängu laadimiseks.

Neljas ühenduspunkt on aadressil '/*games*/{*gamePlayId*}', mille kaudu tehakse POST päringuid kasutaja loodud lahenduse saatmiseks ning nende põhjal loodava animatsiooni näitamiseks vajalike andmete tagasi saamiseks. Lahendus lisatakse päringule loendina IBox liidese objektidest, mis omakorda sisaldavad ka kasutaja lisatud alamkaste ning lüngatäiteid. Joonisel 14 oleva liidese objektide loendi põhjal kontrollib tagarakendus algoritmi õigsust ning genereerib selle toimimist kujutava tegevuste jada koos lõpptulemuse teatega.

```
export interface IBox {
	id: string;
	name: string;
	isParent: boolean;
	innerBoxes: IBox[];
	gapFillers: IGapFiller[] | null[] | [];
	idInSolution?: number;
	randomNumber: number;
}
```

Joonis 14. Lahenduskasti liides IBox.

Klientrakendus saab vastusena joonisel 15 toodud ISolutionResult liidesele vastav objekti, mille põhjal luuakse animatsioon.

```
export interface ISolutionResult {
	message: string | null;
	error: string | null;
	isCompleted: boolean | null;
	attemptsLeft: number | null;
	currentLevelNumber: number | null;
	gamePlayId: string | null;
	errorMessage: string | null;
	activities: IActivity[] | null;
	characterPicture: string | null;
	gameObjectPicture: string | null;
	backgroundPicture: string | null;
	obstaclePicture: string | null;
	gameBoard: string | null;
}
```

Joonis 15. Kasutaja lahenduse tulemust iseloomustav liides ISolutionResult.

4.6 Lahendusalgoritmi koostamise võimaluse loomine React DnD abil

React DnD (*Drag and Drop*) on teek keerukate kasutajaliideste loomiseks, mis nõuavad elementide hiirega pukseerimise funktsionaalsust. Eeskätt on see mõeldud rakendustele, kus elemendi ühest komponendist teise lohistamise tagajärjel tuleb transportida ka andmeid ning muuta vastavalt pukseerimisega seotud sündmustele soovitud elementide välimust ja rakenduse olekut [54].

Klientrakenduse nõuetele vastava lahendamisvõimaluse loomiseks muudeti vahenditena etteantud kastid ning lüngatäited lohistatavaks, kasutades vastavates komponentides React DnD *useDrag()* haaki. Igale vahendile pandi kaasa ka objekti andmed, et lohistamise sihtmärk elemendi pillamisel koostatud lahenduse objekti õigesti uuendaks. Joonisel 16 välja toodud lüngatäite komponendi pukseerimise loogika.

```
const [{isDragging}, drag] = useDrag(() => {
	type: ItemTypes.GAP_FILLER,
	collect: (monitor) => ({
		isDragging: monitor.isDragging()
	}),
	item: () => {
		return {
			id: gapFiller.id, name: gapFiller.name,
			isGameObject: gapFiller.isGameObject,
			randomNumber: gapFiller.randomNumber
		}
	},
	end: (item, monitor) => {
		const didDrop = monitor.didDrop();
		if (didDrop === true) {
			removeGapFillerFromSolution();
		}
	}
}))
```

Joonis 16. Lüngatäite komponendi hiirega pukseerimise loogika React DnD abil.

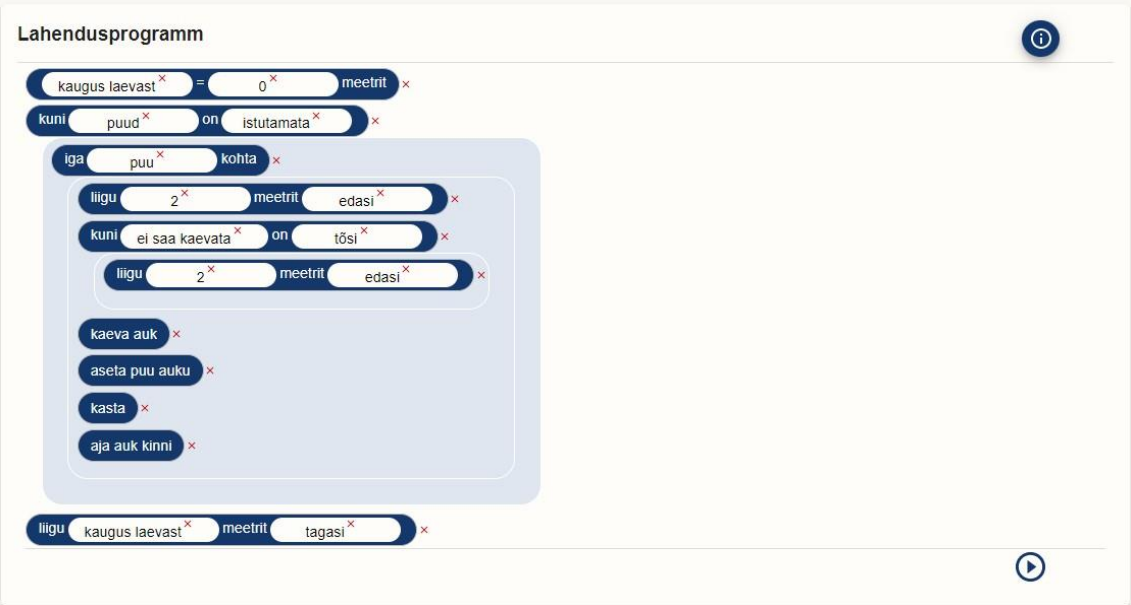
Lohistamise sihtmärkideks tuli erinevate nõuete tõttu määrata neli komponenti. Üldise lahenduse komponent SolutionCard.tsx, et sinna saaks hakata algoritmi looma. Samuti osade kastide alla tekkiv lisaala ChildBoxArea.tsx, kuhu lohistada tegevused, millele tahetakse vastavat tingimust rakendada. Lüngatäidete lünkadesse lisamiseks oli vajalik luua soovitud tüüpi elementide pillamisvõimalus komponenti Gap.tsx ning lahenduses olevate kastide ümberjärjestamise funktsionaalsuse tekitamiseks tehti lohistamise sihtmärkideks ka algoritmi lohistatud kastid. See saavutati React DnD *useDrop()* haagiga, milles määrati aktsepteeritud vastuvõetava elemendi tüüp, saadavad andmed ning soovitav käitumine, kui element on vastavasse komponenti pillatud.

Joonis 17 kujutab lahendusvahenditest lohistatud kasti vastuvõtmist komponendis SolutionCard.tsx. Kast lisatakse käimasoleva mängu lahendusele veebilehitseja sessiooni mälus, tänu millele on võimalik lahendust säilitada ka lehe värskendamisel või mängust lahkumisel, hoidmata seda andmebaasis. Viimasel juhul oleks lehe jõudlus oluliselt häiritud, kuna iga lahenduses tehtava muudatuse – millegi lisamise, kustutamise või ümberjärjestamise puhul tuleks teha päring andmebaasi. Kasutajaliidese uuendamiseks värskete andmetega muudetakse ka komponendi olekut.

```
const [{ isOver }, drop] = useDrop(() => ({
        accept: ItemTypes.BOX,
        drop: (item: {
            boxId: string, boxName: string, gapFillers: IGapFiller[],
            innerBoxes: IBox[], isParent: boolean, randomNumber:
             number}, monitor) => {
            let solutionBoxes =
       getSolutionBoxesFromSessionStorage(props.gameId);
            if (hasInitializedCorrectSolution === false) {
                setHasInitializedCorrectSolution(true);
            }
            if (monitor.isOver()) {
                let droppedBox: IBox = { id: item.boxId, name:
                    item.boxName, gapFillers: item.gapFillers,
                    innerBoxes: item.innerBoxes, isParent:
                    item.isParent, idInSolution: Math.random(),
                    randomNumber: item.randomNumber
                };
                if (solutionBoxes.length > 0) {
                    solutionBoxes.push(droppedBox);
                } else {
                    solutionBoxes = [droppedBox];
                }
                sessionStorage.setItem("solutionBoxesForGame" +
                    props.gameId, JSON.stringify(solutionBoxes));
                setBoxes(solutionBoxes);
            }
        },
        collect: monitor => ({
            isOver: !!monitor.isOver(),
        }),
    }), []);
```

Joonis 17. Pillatava kasti vastuvõtmine lahenduse komponendis React DnD abil.

Kirjeldatud metoodikat kombineerides loodi funktsionaalsus, mis võimaldab luua kasutaja soovitud algoritme. Tingimuslausete ning iteratiivsete kastidega saab koostada piiramatu sügavusega lahenduse, mille igasse kihti on võimalik lisada kuitahes palju tegevusi. Joonisel 18 on näha fiktiivne algoritm, demonstreerimaks kasutajaliidese võimalusi.



Joonis 18. Fiktiivne näidisalgoritm ülesande lahendamise vaates.

4.7 Sisendile vastava animatsiooni loomine Anime.js teegi abil

Anime.js on teek, mis võimaldab API kaudu kontrollida veebilehe valitud elementide CSS omadusi ja DOM-atribuute [31]. Teek lisati sõltuvusena projekti package.json faili, manipuleerimaks vastavalt serveripoolsest teenusest saadud sisendile genereeritud HTML-elemente.

Animatsiooni loomiseks on vajalik, et oleks teada iga lehel paikneva animeeritava objekti identifikaator või klass. Kuna erinevatel mängudel võivad animeeritavad tegevused olla erineva sisuga, on lehe elemendid ning animatsiooni ajajoon genereeritud vastavalt tagarakendusest saabunud muutujate väärtustele. Joonis 19 kujutab pildielementide genereerimist serveripoolsest rakendusest saabuvate veebiaadresside põhjal. Pildid on vajalikud mängu tegelase sooritatavate tegevuste näitamiseks.

```
solutionResult.activities!.map((activity) => {
      if (activity.functionType === "special" && activity.picture &&
                  !actionUrls.includes(activity.picture)) {
            actionUrls.push(activity.picture)
            return (<img key={activity.picture}
                        src={activity.picture}
                        alt="specialActionPicture"
                        className="specialActionPicture"
                        id={activity.picture} />
            )
      }
})
```

Joonis 19. Animeeritavate pildielementide genereerimine vastavalt sisendile.

tALGOtech rakenduses näidatava animatsiooni eesmärk on kasutajale ette mängida tema loodud lahenduse tulemus. See eeldab kindla alguse ja lõpuga lineaarset tegevuste jada, mistõttu põhineb animatsioon Anime.js-i ajajoonel, kuhu tekitatakse vastavalt sisendile loend vajalike elementide käitumisest. Joonisel 20 lisatakse tegevuste ning neid iseloomustavate tekstide soovitav käitumine animatsiooni ajajoonele. Juhul, kui tegevusel on ka lisatagajärg – lõputöö käigus välja arendatud mängus peab puu selle auku asetamisel nähtavale ilmuma, lisatakse see samuti animatsioonile.

```
else if (activity.functionType === "special") {
                    tl.add({
                        targets: activity.picture ?
[document.getElementById(activity.picture),
document.getElementById(activity.name)] :
document.getElementById(activity.name),
                        opacity: 100
                    });
                    if (activity.animateGameObject === true) {
                        tl.add({
                            targets:
document.getElementById(characterLocationOnGameBoard.toString()),
                            opacity: 100
                        })
                    }
                    tl.add({
                        targets: activity.picture ?
[document.getElementById(activity.picture),
                        document.getElementById(activity.name)] :
document.getElementById(activity.name),
                        opacity: 0
                    });
}
```

Joonis 20. Tegevuste ja kirjeldavate tekstide käitumise lisamine animatsiooni ajajoonele.

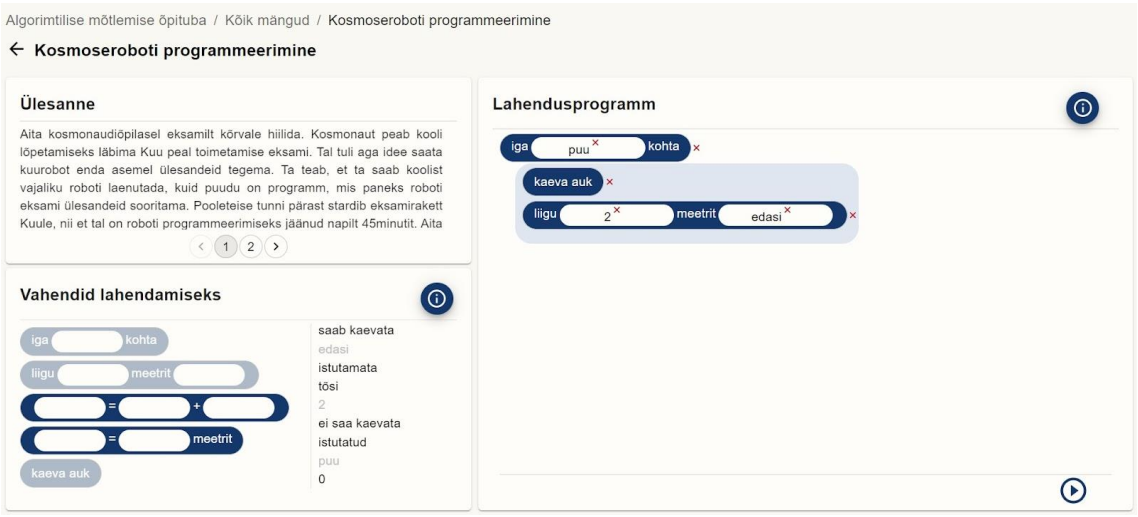
Joonisel 21 on hetk valmis arendatud mängu animatsioonist, kus tegelane on parasjagu puud istutamas.

Joonis 21. Animatsioonivaade puu istutamisest.

4.8 Raskusastmete erisused kasutajaliideses

Igas mängus on kolm raskusastet, mille hulgast kuvatakse kasutajale esmalt keskmine. Kõigi tasemete lahendamiseks on kasutajale ettenähtud teatud arv katseid, mis vähenevad algoritmi käivitamisel animeeritud tulemuse vaatamiseks. Kui ülesannet ei õnnestunud katsete lõppemise hetkeks lahendada, suunatakse mängija edasi kergemale tasemele, eduka lahenduse puhul aga raskemale. Klientrakendusse jõuab informatsioon mängija allesjäänud katsete ning hetkese taseme kohta koos mängu ülejäänud andmetega, mille järgi tekitatakse kasutajaliidesesse vajalikud erisused.

Igal tasemel on lahendamise vaates etteantud vahendite juures lisainfo nupp, millele vajutades avaneb lahendusvahendite reeglistikku kirjeldav tekst. Keskmisel ja raskel tasemel pole teada, mitu korda mingit vahendit vaja läheb. Kergel tasemel on kõike etteantud täpselt vajaminev hulk, mistõttu muudetakse kõik kasutaja lahendusse lisatud vahendid desaktiveerituks. Joonisel 22 on kasutaja alustanud kergel tasemel lahenduse koostamist, mistõttu on juba algoritmi paigutatud vahendid desaktiveeritud.



Joonis 22. Pooleliolev lahendus mängu kergel tasemel.

Samuti on kasutaja jaoks silmnähtavaks erisuseks takistuste lisandumine mängu raskel tasemel. Need ei luba kasutajal teatud tegevusi sooritada, mistõttu tuleb enne nende tegemist lisada lahendusse sobivate tingimuste kontroll. Lahenduse vaates erisusi ei teki, kuid algoritmi tööd kujutavasse animatsiooni ilmuvad tagarakenduses genereeritud mängulauale juhuslikele positsioonidele takistusi kujutavad pildid.

4.9 Serverrakenduse arendus

Klientrakendusega arendati samaaegselt ka tALGOtech serverrakendust. Spring Boot raamistikus arendatud rakenduse projekti algseadistustena kasutati projekti ehitamise tööriistaks Gradle'it, programmeerimiskeeleks Javat ja raamistikuks Spring Booti versiooni 2.6.4. Ühtlasi lisati vajalikud teegid. Rakenduse ülesehitus koosneb äriloogikast, kontrolleritest, andmeedastuskihist, mängude loogikast ja andmemudelitest. Arendusprotsessi jooksul kasutati mugavuse tõttu andmebaasina HyperSQL Database-i, kuid lõpufaasis ühendati rakendus Docker tööriista kasutades MariaDB andmebaasiga. Selle edasise haldamise lihtsustamiseks võeti kasutusele ka andmebaaside jaoks mõeldud versioonihalduse teek Flyway. Serverrakenduse arenduse kohta võib täpsemalt lugeda Maarja Helena Elisabeth Hoopi lõputööst.

5 Hinnang loodud klientrakendusele

Lõputöö raames valminud klientrakendust hinnatakse eelkõige saavutatud funktsionaalsuse alusel, vaadeldes vastavust analüüsi käigus välja selgitatud nõuetele. Lisaks on oluline visuaalne välimus ning klientrakenduse kasutajamugavus, mille alla kuulub ka päringute kui kulukate operatsioonide optimeerimine. Analüüsitakse ka tehnoloogiliste valikute õnnestumist, mille puhul hinnatakse vahendite uudsust ja sobivust käesoleva klientrakenduse teostamiseks.

5.1 Kasutatavus

Valminud klientrakendus vastab kõigile tava- ehk registreerimata kasutaja rolliga seotud funktsionaalsetele ja mittefunktsionaalsetele nõuetele. Tänu sellele on võimalik klientrakendus kasutusele võtta Tallinna Tehnikaülikooli IT Kolledži erialasid tutvustavate töötubade läbiviimiseks. Töötoa läbiviijatele, kes ei pruugi rakendusega eelnevalt tuttavad olla, kuid peavad oskama kasutajate potentsiaalsetele küsimustele vastata, on koostatud ka abistav juhend, mille võib leida lisast 4.

Kasutajaliidese ülesehitus vastab üldjoontes analüüsi käigus loodud prototüübile, kuid sai mugavama kasutajakogemuse loomiseks teatud täiendusi. Lahendusse lohistamisel ilmuvad elementide kõrvale kustutamiseks mõeldud vastavad ikoonid ning enda alla alamtegevusi koondavate kastide juurde selgelt piiritletud lisaalad. Alamtegevuste lisamisest tekkiva kihistunud lahenduse sügavus on piiramatu.

Kasutaja lahenduse põhjal loodav animatsioon genereeritakse vastavalt sisendile, võttes arvesse serveripoolsest rakendusest saadetavaid andmeid näidatavate tegevuste ning selleks mõeldud kujutiste kohta. See tagab animeeritud tagasiside tulevikus loodavatele mängudele, ilma et oleks vaja midagi käsitsi juurde lisada või kohandada.

Klientrakendusele tavakasutajaga seotud funktsionaalsuste teostamise ootamatu keerukuse ning lõputöö ajalise piiratuse tõttu jäid täitmata administraatori rolliga seotud nõuded. Ülejäänud kasutajaliidesega võrreldes on tegu aga pigem standardlahendusega. Saavutamata jäänud funktsionaalsus ei takista klientrakenduse kasutuselevõttu algoritmilise mõtlemise töötubade läbiviimiseks.

5.2 Kasutatud tehnoloogiate uudsus ja sobivus

Tehnoloogiate valikul võeti arvesse nende populaarsust, ühilduvust teiste kasutatavate vahenditega, autori kogemust ja pakutava funktsionaalsuse vastavust klientrakenduse vajadustele. Arendusraamistik React, millest kasutati arenduse alustamise hetkeks uusimat avalikustatud versiooni 17.0.2, osutus kõigi kriteeriumite suhtes sobivaks. Samuti vastas ootustele väga põhjaliku ja hästiorganiseeritud dokumentatsiooniga MUI disainiteek. Animatsiooni loomiseks kasutatud Anime.js pakkus valminud klientrakenduse vajadustele vastavat ajajoone-lahendust, mis võimaldas koos laia kasutajabaasi ning dokumentatsiooniga oodatust kergemat soovitud tulemuseni jõudmist. Arenduse käigus selgunud vajadus hiirega pukseerimise abiteegi järele lisas projektile ka React DnD sõltuvuse, mille kasutamaõppimine oli alguses küll keeruline ja aeganõudev, kuid tänu millele täideti lõpuks kõik kasutaja lahenduskogemusele seatud funktsionaalsed nõuded.

Arenduseks valitud tehnoloogiad olid kõik aktuaalsed ning vastasid klientrakenduse vajadustele, mistõttu jäädi tehtud otsustega igati rahule.

5.3 Võimalused edasiseks arenduseks

Suurima prioriteediga edasiarenduseks on peamiselt administraatori rolliga seotud püstitatud nõuete teostamine. Selle jaoks tuleb luua vajalike päringute tegemise võimalus ning vaated sisselogimiseks ja mängude kuvamiseks, mis sisaldaks omakorda CRUD funktsionaalsust. Nii saaks administraator rakenduse mängude nimekirja täiendada ja hallata, ilma et oleks vaja otsest kokkupuudet koodiga.

Pikemaajalise eesmärgina nähakse rakenduse potentsiaali areneda registreeritud kasutajatega algoritmilise mõtlemise mängude lahendamise ning loomise keskkonnaks, kus kasutajatel on võimalik ise soovitud temaatika ning ülesandepüstitustega mänge juurde luua. Tänu loovuslikule aspektile kõrgendaks see veelgi huvi algoritmilise mõtlemise vastu ja aitaks hakata õpitut uues kontekstis rakendama. Kasutajatel oleks pärast administraatori kinnituse saamist võimalik enda loodud mänge ka avalikustada.

Rakendusele rahvusvahelise mastaabi andmiseks tuleks lisada keelevaliku võimalus, et tutvustada IT Kolledži erialasid ka potentsiaalsetele välistudengitele.

tALGOtech klientrakenduse kood on Tallinna Tehnikaülikooli töötajatele ja tudengitele avatud, et seda saaksid tulevikus edasi arendada ka teised üliõpilased. Koodihoidla link on toodud lisas 5. Sealoleva projekti README.md failis on olemas ka samm-sammuline juhend soovitavate edasiarenduste teostamiseks ning viide käesolevale lõputööle kui projekti dokumentatsioonile.

6 Kokkuvõte

Lõputöö eesmärgiks oli koostöös kaasautoriga luua veebirakendus Tallinna Tehnikaülikooli IT Kolledžile algoritmilise mõtlemise töötubade läbiviimiseks. Käesoleva töö autor planeeris ning teostas klientrakenduse, mis täidaks olemasolevate alternatiivide puudujääke. Rakenduse esialgsesse skoopi kuulusid mängu lahendamise ja animatsiooni genereerimise väljaarendamine ning mängude haldamise võimaluse loomine administraatorile. Viimane jäi aga rakenduse keerukuse ning töö ajalise piirangu tõttu lõplikkust skoobist välja.

Töö analüüsis seati loodavale rakendusele kindlad nõuded ning võrreldi erinevate tehnoloogiate sobivust nendele vastava klientrakenduse teostamiseks. Kasutajakogemuse disaini planeerimine ning prototüübi tegemine lihtsustas oluliselt arendusprotsessi, kuna tagas tulevase rakenduse detailideni läbimõtlemise. Arendusprotsessi kirjeldavas peatükis anti ülevaade klientrakenduse tähtsamate osade tööpõhimõtetest, kasutatud tehnoloogiate rakendamisest ja suhtlusest serverrakendusega.

Lõputöö loetakse õnnestunuks, kuna valminud klientrakendus on eesmärgipäraselt kasutatav. tALGOtech rakendus on ligipääsetav TalTech IT Kolledži sisevõrgus ning selle link on toodud lisas 6. Töö mahukuse tõttu välja arendamata jäänud funktsionaalsused on selgelt sõnastatud ning arenduse käigus on silmas peetud nende võimalikult mugavat lisamist tulevikus. Tänu projekti kergesti hallatavale struktuurile, põhjalikule dokumentatsioonile käesoleva lõputöö näol ning avatud koodibaasile saavad edaspidi klientrakendusse küllaltki kerge vaevaga panustada ka teised arendajad.

Kasutatud kirjandus

[1] The Bowers Institute, „The Tech Interactive,“ [Võrgumaterjal]. Available: https://www.thetech.org/sites/default/files/techtip_computationalthinking_v3.pdf. [Kasutatud 13.03.2022].

[2] P. Lorents, Keel ja loogika, Tallinn: ESTONIAN BUSINESS SCHOOL, 2000.

[3] N. G. K. V. Y. S. I. S. L. Albina A. Temerbekova, „CEUR Workshop Proceedings,“ [Võrgumaterjal]. Available: http://ceur-ws.org/Vol-2834/Paper36.pdf. [Kasutatud 13.03.2022].

[4] „Medium,“ 03 12 2018. [Võrgumaterjal]. Available: https://medium.com/tech-based-teaching/if-curious-then-learn-a-brief-intro-to-algorithmic-thinking-ba683bf44994. [Kasutatud 13.01.2022].

[5] „About Scratch,“ Scratch, [Võrgumaterjal]. Available: https://scratch.mit.edu/about. [Kasutatud 14.03.2022].

[6] „Scratch for Educators,“ Scratch, [Võrgumaterjal]. Available: https://scratch.mit.edu/educators/. [Kasutatud 14.03.2022].

[7] „Explore,“ Scratch, [Võrgumaterjal]. Available: https://scratch.mit.edu/explore/projects/games/. [Kasutatud 14.03.2022].

[8] „What is Hopscotch?,“ Hopscotch, [Võrgumaterjal]. Available: https://help.gethopscotch.com/article/5-what-is-hopscotch. [Kasutatud 14.03.2022].

[9] „The Pack,“ nysci, [Võrgumaterjal]. Available: https://nysci.org/the-pack. [Kasutatud 14.03.2022].

[10] „Autothinking,“ [Võrgumaterjal]. Available: https://sites.google.com/view/autothinking-estonian/avaleht?authuser=0. [Kasutatud 14.03.2022].

[11] „Top 10 Web Development Stacks you need to look out for in 2022,“ Octal IT Solution, [Võrgumaterjal]. Available: https://www.octalsoftware.com/blog/best-web-development-stacks. [Kasutatud 16.02.2022].

[12] Vinugayathri, „PHP Vs Javascript: The Right Tech For Your Next Big Project,“ Clarion Technologies, [Võrgumaterjal]. Available: https://www.clariontech.com/blog/php-vs-javascript-the-right-option-for-your-next-big-project. [Kasutatud 21.04.2022].

[13] A. Azzam, „TypeScript vs. JavaScript: Why you should switch to TypeScript,“ Medium, [Võrgumaterjal]. Available: https://medium.com/weekly-webtips/typescript-vs-javascript-why-you-should-switch-to-typescript-31654de9a664. [Kasutatud 21.04.2022].

[14] L. S. Vailshery, „Most used web frameworks among developers worldwide, as of 2021,“ Statista, 23.02.2022. [Võrgumaterjal]. Available: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/. [Kasutatud 25.02.2022].

[15] „React,“ React, [Võrgumaterjal]. Available: https://reactjs.org/. [Kasutatud 15.02.2022].

[16] „jQuery,“ jQuery, [Võrgumaterjal]. Available: https://jquery.com/. [Kasutatud 15.02.2022].

[17] „Angular,“ Angular, [Võrgumaterjal]. Available: https://angular.io/. [Kasutatud 15.02.2022].

[18] Angular, „Features & Benefits,“ Angular, [Võrgumaterjal]. Available: https://angular.io/features. [Kasutatud 15.02.2022].

[19] Vue.js, „Vue.js - The Progressive JavaScript Framework,“ Vue.js, [Võrgumaterjal]. Available: https://vuejs.org/. [Kasutatud 15.02.2022].

[20] R. A. S. L. Daniel Roth, „Overview to ASP.NET Core,“ Microsoft, 26.03.2022. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0. [Kasutatud 21.04.2022].

[21] „Google Trends võrdlus: Vue.js vs. React,“ Google, [Võrgumaterjal]. Available: https://trends.google.com/trends/explore?cat=31&q=Vue.js,React. [Kasutatud 21.04.2022].

[22] K. Varkki, „The Most Popular React UI Component Libraries in 2022,“ SitePoint, 13.12.2021. [Võrgumaterjal]. Available: https://www.sitepoint.com/popular-react-ui-component-libraries/. [Kasutatud 16.02.2022].

[23] MUI, „MUI: The React component library you always wanted,“ MUI, [Võrgumaterjal]. Available: https://mui.com/. [Kasutatud 15.04.2022].

[24] React-Bootstrap, „React-Bootstrap,“ [Võrgumaterjal]. Available: https://react-bootstrap.github.io/. [Kasutatud 17.02.2022].

[25] Ant Design, „Ant Design of React,“ Ant Design, [Võrgumaterjal]. Available: https://ant.design/docs/react/introduce. [Kasutatud 17.02.2022].

[26] B. Holland, „11 JavaScript frameworks for creating graphics,“ Creative Bloq, 19.11.2020. [Võrgumaterjal]. Available: https://www.creativebloq.com/features/amazing-graphical-javascript-frameworks. [Kasutatud 16.02.2022].

[27] Three.js, „Fundamentals,“ Three.js, [Võrgumaterjal]. Available: https://threejs.org/manual/#en/fundamentals. [Kasutatud 16.02.2022].

[28] GreenSock, „What's so special about GSAP?,“ GreenSock, 17 01 2014. [Võrgumaterjal]. Available: https://greensock.com/why-gsap. [Kasutatud 16.02.2022].

[29] Two.js, „Two.js,“ Two.js, [Võrgumaterjal]. Available: https://two.js.org/. [Kasutatud 16.02.2022].

[30] Pts, „Get started,“ Pts, [Võrgumaterjal]. Available: https://ptsjs.org/guide/get-started-0100. [Kasutatud 16.02.2022].

[31] Anime.js, „Anime.js documentation,“ Anime.js, [Võrgumaterjal]. Available: https://animejs.com/documentation/. [Kasutatud 15.02.2022].

[32] PixiJS, „PixiJS,“ [Võrgumaterjal]. Available: https://pixijs.com/. [Kasutatud 16.02.2022].

[33] Zdog, „What is Zdog?,“ [Võrgumaterjal]. Available: https://zzz.dog/#what-is-zdog. [Kasutatud 16.02.2022].

[34] Snap.svg, „Why Snap?,“ Snap.svg, [Võrgumaterjal]. Available: http://snapsvg.io/about/. [Kasutatud 16.02.2022].

[35] Bundlephobia, „Anime.js bundle size,“ Bundlephobia, [Võrgumaterjal]. Available: https://bundlephobia.com/package/animejs@3.2.1. [Kasutatud 18.02.2022].

[36] Software Testing Help, „Comparing Top Source Code Management Software,“ Software Testing Help, 03.04.2022. [Võrgumaterjal]. Available: https://www.softwaretestinghelp.com/best-source-code-management-tools/#Comparing_Top_Source_Code_Management_Software. [Kasutatud 21.04.2022].

[37] GitHub, „GitHub,“ GitHub, [Võrgumaterjal]. Available: https://github.com/. [Kasutatud 20.02.2022].

[38] Bitbucket, „Bitbucket | The Git solution for professional teams,“ Atlassian, [Võrgumaterjal]. Available: https://bitbucket.org/product. [Kasutatud 20.02.2022].

[39] Bitbucket, „Bitbucket Cloud vs. GitHub,“ Atlassian, [Võrgumaterjal]. Available: https://bitbucket.org/product/comparison/bitbucket-vs-github. [Kasutatud 20.02.2022].

[40] GitLab, „Is it any good? | GitLab,“ GitLab, [Võrgumaterjal]. Available: https://about.gitlab.com/is-it-any-good/. [Kasutatud 20.02.2022].

[41] GitLab, „Unify the entire DevOps lifecycle with GitLab,“ GitLab, [Võrgumaterjal]. Available: https://about.gitlab.com/stages-devops-lifecycle/. [Kasutatud 20.02.2022].

[42] InterviewBit, „9 Best JavaScript IDE & Source Code Editors [2021],“ 18.10.2021. [Võrgumaterjal]. Available: https://www.interviewbit.com/blog/javascript-ide/. [Kasutatud 20.02.2022].

[43] JetBrains, „WebStorm: Features,“ JetBrains, [Võrgumaterjal]. Available: https://www.jetbrains.com/webstorm/features/. [Kasutatud 20.02.2022].

[44] Visual Studio Code, „Why Visual Studio Code?,“ Visual Studio Code, [Võrgumaterjal]. Available: https://code.visualstudio.com/docs/editor/whyvscode. [Kasutatud 20.02.2022].

[45] K. Bracey, „What Is Figma?,“ 26.11.2018. [Võrgumaterjal]. Available: https://webdesign.tutsplus.com/articles/what-is-figma--cms-32272. [Kasutatud 19.02.2022].

[46] Create React App, „Creating a TypeScript app,“ Facebook, Inc, [Võrgumaterjal]. Available: https://create-react-app.dev/docs/getting-started/#creating-a-typescript-app. [Kasutatud 14.04.2022].

[47] json-server, „GitHub - typicode/json-server,“ [Võrgumaterjal]. Available: https://github.com/typicode/json-server. [Kasutatud 14.04.2022].

[48] React, „Thinking in React,“ React, [Võrgumaterjal]. Available: https://reactjs.org/docs/thinking-in-react.html?fbclid=IwAR2Y8Nj2uyA8f8F_6nMu5A4tT5xaNiaTPcOTh2SsJROTHY6F6DzxwE-scwI. [Kasutatud 15.04.2022].

[49] P. Bolmér, „How To Implement a Single Page Application Using React-Router,“ Better Programming, 10.08.2021. [Võrgumaterjal]. Available: https://betterprogramming.pub/how-to-implement-a-single-page-application-using-react-router-cc6b9e5c3aec. [Kasutatud 14.04.2022].

[50] Create React App, „Folder Structure,“ [Võrgumaterjal]. Available: https://create-react-app.dev/docs/folder-structure/. [Kasutatud 17.04.2022].

[51] React, „Rendering Elements,“ [Võrgumaterjal]. Available: https://reactjs.org/docs/rendering-elements.html. [Kasutatud 17.04.2022].

[52] npm Docs, „package.json | npm Docs,“ [Võrgumaterjal]. Available: https://docs.npmjs.com/cli/v7/configuring-npm/package-json#description. [Kasutatud 15.04.2022].

[53] Axios, „Getting Started | Axios Docs,“ [Võrgumaterjal]. Available: https://axios-http.com/docs/intro. [Kasutatud 14.04.2022].

[54] React DnD, „React DnD - about,“ [Võrgumaterjal]. Available: https://react-dnd.github.io/react-dnd/about. [Kasutatud 17.04.2022].

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

[56] P. Bolmér, „How To Implement a Single Page Application Using React-Router,“ Better Programming, 10.08.2021. [Võrgumaterjal]. Available: https://betterprogramming.pub/how-to-implement-a-single-page-application-using-react-router-cc6b9e5c3aec. [Kasutatud 14.04.2022].

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

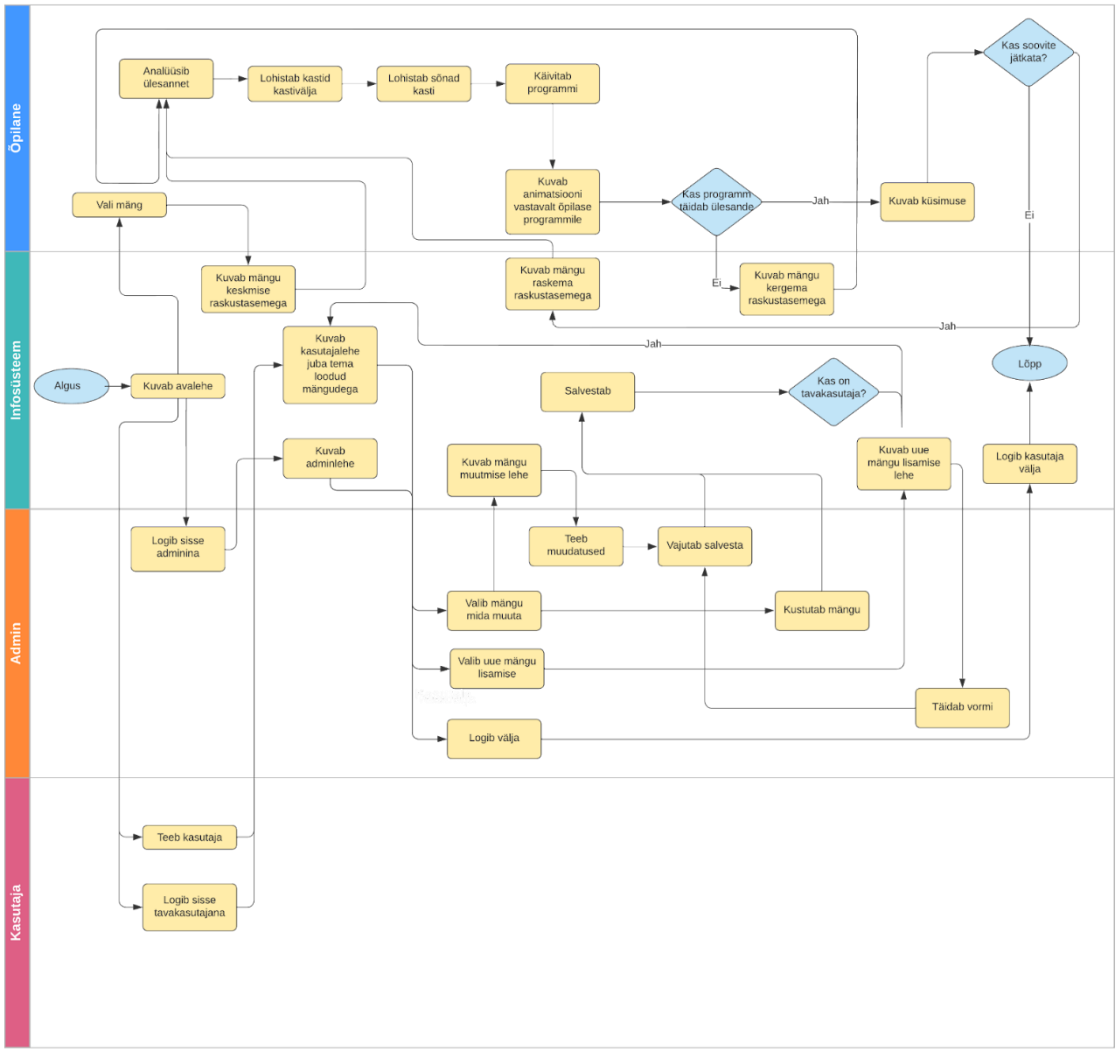
Mina, Triinu Malv

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Klientrakenduse loomine algoritmilist mõtlemist arendavale TalTechi veebimängule tALGOtech“, mille juhendaja on Meelis Antoi ja abijuhendaja on Kerman Saapar.
   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.

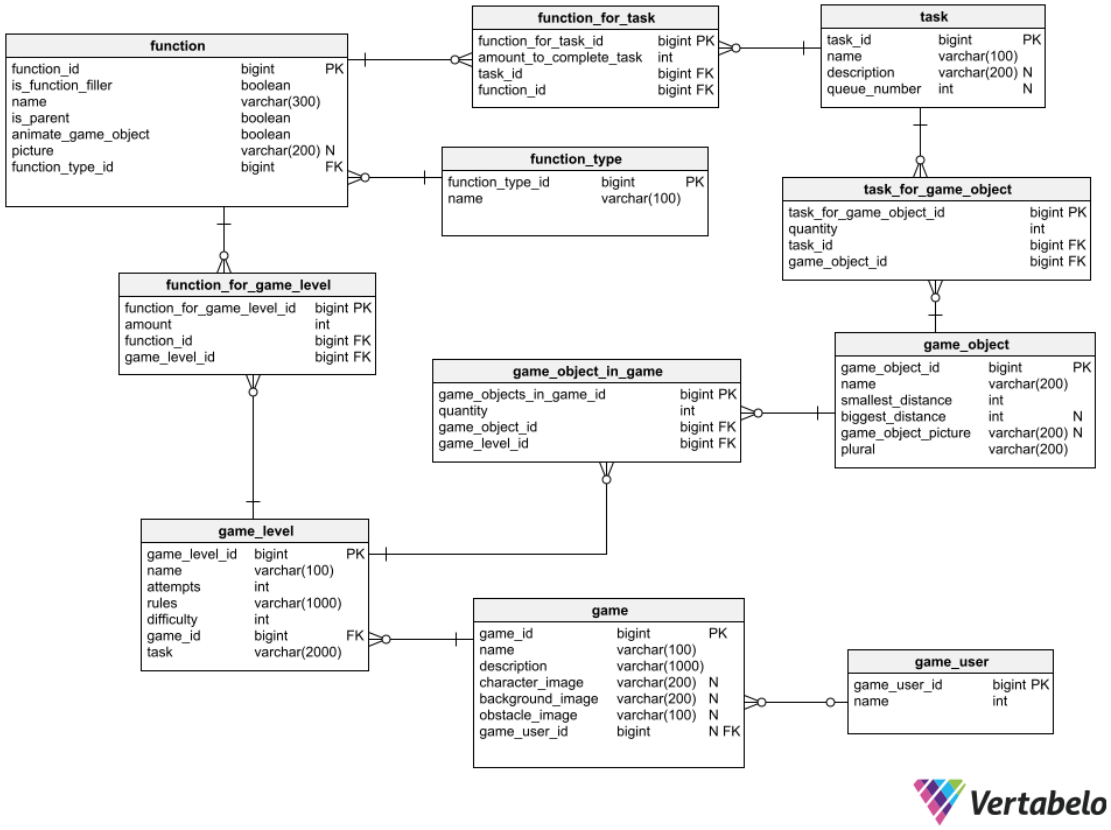
15.05.2022

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 – tALGOtech infosüsteemi skeem



Lisa 3 – Andmebaasi täielik olemi-suhte diagramm



Lisa 4 – tALGOtech kasutusjuhend töötoa läbiviijale

Lahenduse koostamine

tALGOtech rakenduse mängudes esitatud ülesannete lahendamiseks tuleb kasutajal etteantud vahenditest moodustada sobiv algoritm. Vahendid jagunevad kaheks: kastid ja lüngatäited. Algoritmi koostamiseks tuleb kastid õigesti järjestada ning osades kastides olevad lüngad vajalike sõnade või numbritega täita.

Rakenduse kasutajad leiavad lühiversiooni järgnevatest juhistest, vajutades ülesande lahendamise vaates olevatele infonuppudele (üks on etteantud vahendite ning teine lahendamisala juures).

Kastide järjestamine

- Kastides kirjeldatud tegevuste järjestamiseks tuleb need **vahendite hulgast lahendusalasse nihutada**.
- Mõned kastid seisnevad teatud tingimuse seadmises (näiteks “Kui saab istutada”) või korduste tekitamises (näiteks “Kuni puud on istutamata”). Nende alla tekivad pärast lahendusse nihutamist **helesinised lisaalad**, kuhu on võimalik paigutada **nende kastide mõjualasse kuuluvad tegevused**.
- Lahenduses olevaid kaste on võimalik omavahel **ümber järjestada**, kuid seda saab teha vaid n-ö sama kihi siseselt. See tähendab, et omavahel on võimalik ümber järjestada kas lahenduse põhialas olevaid kaste või samas lisaalas olevaid kaste.
- Kaste on võimalik lahendusest **eemaldada**, vajutades kasti kõrval olevat ristikest.
- Selleks, et kasti **ühest kihist teise liigutada**, tuleb kast lahendusest kustutada ning vahendite hulgast uuesti soovitud kohale nihutada.

Lüngatäidete kasutamine

- Lüngatäite **soovitud lünka lisamiseks** tuleb see vahendite hulgast juba lahenduses oleva kasti lünga kohale nihutada.
- Lahenduses olevaid lüngatäiteid on võimalik **ühest lüngast teise** nihutada.
- Lahenduses olevate lüngatäidete **eemaldamiseks** tuleb vajutada lüngatäite kõrval olevat ristikest.

Tasemete erisused

- **Kergel tasemel** on etteantud **täpselt vajaminev hulk vahendeid** ehk iga kasti ja lüngatäidet läheb vaja ainult üks kord. Selle tagamiseks muutuvad juba lahendusse nihutatud vahendid deaktiveerituks. Kui kasutaja mõne vahendi algoritmist kustutab, muutub see taas aktiivseks.

- **Keskmisel ja raskel tasemel pole teada**, kui **mitu korda iga vahendit** õige lahenduse koostamiseks **vaja läheb**. Seega jäävad vahendid pärast kasutamist aktiivseks, kuna sama vahendit võib algoritmi loomiseks vaja minna mitmel korral.

Algoritmi käivitamine

Kasutaja saab koostatud algoritmi toimimist näha, **vajutades** lahendusala alumises osas olevale ***play*-nupule**. Seeläbi **vähendatakse kasutaja allesjäänud katsete arvu** ning avatakse animatsiooni näitav lisaaknake. **Animatsioon** näitab kõiki tegevusi, mida kasutaja algoritm on juhendanud mängu tegelast tegema. Selle lõppedes näidatakse kasutajale, **kas algoritm lahendas ülesande või mitte** ning annab ebaõnnestumise korral ka vihje, mis valesti läks. **Kui algoritm täitis ülesande tingimusi**, saab kasutaja soovi korral **liikuda raskemale tasemele**. Vastasel juhul tuleb kasutajal kas **uuesti proovida** või kui tal praegusel tasemel katsed otsa on saanud, peab ta **kergemale tasemele liikuma**.

Mäng "Kosmoseroboti arendamine"

Mängus on tegelaseks kosmonaudist õpilane, kes peab Kuu peal eksami läbima. Eksami sisuks on ploomipuude istutamine, et luua Kuule puuviljaaeda. Mängu läbimiseks peab mängija kirjutama programmi, mille abil oskaks kosmonaudi robot vastavalt ülesande nõuetele ploomipuid istutada.

Mängu üldkirjeldus: "Aita kosmonaudiõpilasel eksamilt kõrvale hiilida. Kosmonaut peab kooli lõpetamiseks läbima Kuu peal toimetamise eksami. Tal tuli aga idee saata kuurobot enda asemel ülesandeid tegema. Ta teab, et ta saab koolist vajaliku roboti laenutada, kuid puudu on programm, mis paneks roboti eksamiülesandeid sooritama. Pooleteise tunni pärast stardib eksamirakett Kuule, nii et tal on roboti programmeerimiseks jäänud napilt 45 minutit. Aita kosmonaudil "spikerdada"!"

Mängu põhiobjekt on ploomipuu, mida võib istutada minimaalselt iga 2m tagant. Mängu algoritmi koostamiseks on õpilasel võimalik kasutada kaste ja lüngatäiteid. Ploomipuu istutamise erifunktsioonikastid on "augu kaevamine", "puu auku panemine", "augu kinniajamine" ja "kastmine". Kõiki tegevusi tuleb ühe ploomipuu istutamiseks teha üks kord. Lisaks erikastidele on mängus tüüpilised kastid liikumiseks, tsükli loomiseks ja vajalike tingimuste kontrolliks.

Võimalikud kastid:

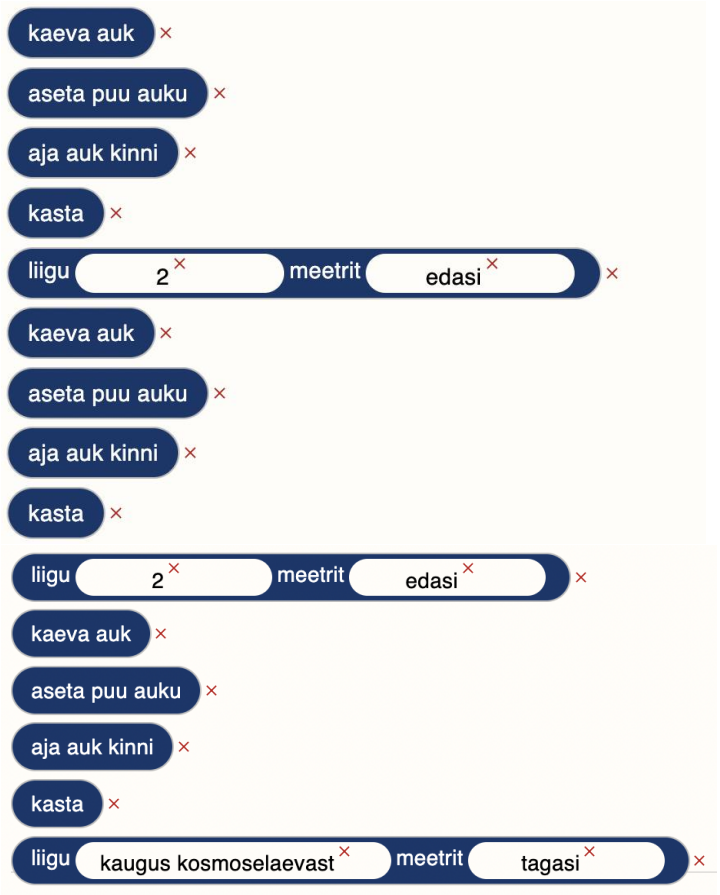
Nimi	**Mida see teeb?**	**Muutujad**
Liigu _ meetrit _	Liikumine	Teepikkus, suund
Kuni _ on istutamata	Tsükkel. Teeb tsükli sisse pandud tegevusi nii mitu korda kui palju on muutujana antud mänguobjekte	Mänguobjektid, mille alusel tsükkel tehakse
Erifunktsioonid: kaeva auku, aseta puu auku, aja auk kinni, kasta	Spetsiaalne juhis ploomipuu istutamise tegevuse täitmiseks	-
Kui saab _	Kontrollib kas mingit tegevust saab teha	Tegevus, mida kontrollitakse

Võimalikud lüngatäited:

Nimi	**Mida see teeb?**
Edasi	Näitab liikumise suunda edasi
Tagasi	Näitab liikumise suunda tagasi
Number (nt 2)	Vahemaa, mille võrra liigutakse
Kaugus kosmoselaevast	Kaugus millestki, näiteks kaugus algusest
Kõik ploomipuud	Mänguobjektid, mille alusel tsükkel tehakse
Istutada	“Kui saab _ “ lüngatäide. Kontrollib, kas selles kohas mängulaual saab teha vajalikke toimingud

Mäng “Kosmoseroboti arendamine” lahendused

Lihtsa taseme lahendus:



Keskmise taseme lahendus:

Raske taseme lahendus:

Lisa 5 – tALGOtech klientrakenduse koodihoidla link

https://gitlab.cs.ttu.ee/trmalv/algorithmic_thinking_workshop_client

Lisa 6 – tALGOtech rakenduse link

http://talgotech.itcollege.ee