TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Oskar Anderson 185611IADB

Vabavaralise rakenduse arendamine andmebaasimudelite visualiseerimiseks

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

Autorideklaratsioon

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

Autor: Karl Oskar Anderson

27.12.2022

Annotatsioon

Käesoleva lõputöö eesmärk on luua andmebaasi skeemide visualiseerimist võimaldav veebirakendus, mis täidab autori poolt varasemalt kasutatud QSEE SuperLite programmi puudujääke. Töö raames võrreldakse alternatiivseid programme, aga nende kasutamine andmebaasi skeemide loomiseks osutub funktsionaalsuselt ebapiisavaks, kasutajaliidesest kohmakaks või hinnalt kalliks. Erinevalt alternatiivsetele rakendustele kasutab antud töö jooksul koostatud rakendus andmebaasi skeemide visualiseerimiseks tekstipõhist formaati, mis võimaldab skeemide kujutamise veebilehekülgede koodiblokkide sees. Töö käigus valmib rakendus, millega on võimalik andmebaasi skeeme visualiseerida ja käivitada kasutaja loodud skripte skeemi andmete põhjal väljundi genereerimiseks.

Töö teoreetilises pooles võrreldakse erinevaid olemasolevaid andmebaasi skeemide visualiseerimist võimaldavate rakenduste puudujääke ja võimekust. Edasi seatakse loodavale rakendusele nõuded ja analüüsitakse rakenduse arendamiseks kasutatavat tehnoloogiat.

Töö praktilises pooles kirjeldatakse loodud rakenduse ülesehitust, rakenduse koodis esinevaid koodimustreid, valminud rakendust ja selle edasiarenduse võimalusi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 45 leheküljel, 6 peatükki, 10 joonist, 6 tabelit.

Abstract

Development of Open-Source Software for Visualizing Database Models

The goal of this thesis is to develop a web application for visualizing database schemas, which abolishes the shortcoming QSEE SuperLite program used by the author previously. Alternative database schema visualization software solutions are compared as part of this thesis, but they are either insufficient in terms of functionality, have a clumsy user interface or are expensive. Unlike alternative solutions, the application developed during this work uses a text-based format for visualizing database schemas, which enables representation of the schemas within code blocks of web pages. The application developed in this thesis will enable visualization of database schemas and allow execution of user created scripts to generate output based on schema data based on user needs.

The theoretical part of this work compares the shortcomings and capabilities of various existing database schema visualization applications. After that, general and functional requirements necessary for application development are defined along with analysis of different possible technologies like programming languages, frameworks and library dependencies for developing the application.

The practical part of this work describes the structure of the created application in terms of development process and layout of files, the code patterns and algorithms used in the application code, the finished application itself according to previously defined requirements and possibilities for further development. In addition, it also contains samples of the application user interface.

The thesis is in Estonian and contains 45 pages of text, 6 chapters, 10 figures, 6 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
canvas	HTML element veebis graafika kuvamise võimaldamiseks
CSV	*Comma-separated values*, arvutustabeli fail
DOCX	pakitud dokumendifailivorming
dok.	Dokumentatsioon (autori kasutatud lühend)
ERD, ER skeem	*Entity Relationship Diagram*, olemi-suhte diagramm
FPS	*frames per second*, kaadrisagedus
GIF	*Graphics Interchange Format*, animatsioone toetav rastergraafika failivorming
HTML	*HyperText Markup Language*, veebilehtede märgistuskeel
iframe	HTML element, mis lubab kuvada teise lehekülje sisu
JSON	*JavaScript Object Notation*, andmevahetusformaat
MD, Markdown	Lihtsasti HTML-iks teisendatav märgistuskeel
modal	Veebilehekülje dialoogiaken, mittepahatahtlik hüpikaken
ODBC	*Open Database Connectivity*, andmebaasi juurdepääsu standard
ORM	*Object Relational Mapping*, meetod programmeerimiskeele andmemudeli teisendamine andmebaasi mudeliks
PNG	*Portable Network Graphics*, rastergraafika failivorming
prog.k.	Programmeerimiskeel (autori kasutatud lühend)
RTF	*Rich Text Format*, tekstipõhine dokumendifailivorming
SQL	*Structurted Query Language*, andmebaasi päringukeel
SVG	*Scalable Vector Graphics*, vektorgraafika failivorming
undo/redo	Rakenduse operatsioonide tühistamine/taastamine
WebGL	*Web Graphics Library*, JavaScript API 2D ja 3D graafika läbi *canvas* elemendi kuvamiseks
XML	*Extensible Markup Language*, laiendatav märgistuskeel, andmevahetusformaat

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tarkvaraarenduse üheks oluliseks osaks on projekti dokumenteerimine. Korralikult dokumenteeritud projekt on terviklik, konkreetselt arusaadav, kergesti hallatav ja hästi kasutatav. Tihtipeale sisaldavad erinevad tarkvaraprojektid andmete töötlemist ja andmete struktureeritud kujul andmebaasi salvestamist. Andmebaasi struktuuri mõistmiseks ning visualiseerimiseks kasutatakse andmebaasi ER skeeme, mis kirjeldavad andmebaasi tabeleid ja nendevahelisi suhteid.

Nende skeemide joonistamiseks on erinevaid rakendusi. Hea dokumentatsiooni koostamiseks peaks nende rakenduste kasutamine olema lihtne, kiire ja funktsioonirikas. Paraku on olemasolevad rakendused mõeldud üldiste skeemide koostamiseks ja pakuvad pigem stiili valikuid, mitte andmebaasi modelleerimiseks vajalikke tööriistu või on hinna ja halva kasutajakogemuse tõttu ebameeldivad. Soovitud programm peaks sisaldama võimalust kasutajatel laiendada rakenduse väljundit läbi skriptimise, olema vabavaraline (skriptimise soodustamine), töötama sõltumata platvormist, võimaldama skeemide kujutamist teksti formaadis ja rakenduse enda kasutamine peaks olema lihtne.

Käesoleva bakalaureusetöö eesmärk on luua tasuta kättesaadav vabavaraline veebirakendus, mis võimaldaks ER skeemide koostamist ja sisaldaks endas skriptimiskeelt kasutaja skriptide käivitamiseks. Rakendus peaks pakkuma huvi kõikidele tarkvaraarendajatele, kes soovivad oma andmebaase dokumenteerida ja visualiseerida.

Bakalaureusetöö on jaotatud 4 osaks. Esmalt kirjeldatakse põhjalikumalt antud töö käigus lahendatavat probleemi ja projekti eesmärki. Seejärel võrreldakse olemasolevaid alternatiivseid lahendusi. Järgnevalt defineeritakse rakenduse nõuded, ja selgitatakse parim tehnoloogia rakenduse arendamiseks. Lõpuks kirjeldatakse valminud rakenduse arendusprotsessis kasutatud koodimustreid ja valminud rakendust ennast.

2 Probleemi püstitus ja projekti eesmärk

Järgnevalt selgitatakse hetkeolukorra tausta, kirjeldatakse lõputöös käsitletavat probleemi ja seatakse projekti eesmärk.

2.1 Taust

Varasemalt kasutas autor andmebaasi mudelite visualiseerimiseks TalTech ülikoolis Andmebaasisüsteemide alused õppeaines kasutatud QSEE SuperLite rakendust. Antud lahendus on loodud Leeds Beckett University akadeemiku ja kunagise QSEE Technologies ettevõtte tegevdirektori Dr. Mark Dixon poolt [1]. Antud tööriist on olemuselt Windows operatsioonisüsteemipõhine akadeemiline hülgvara, viimane rakenduse versioon 1.1.2 ilmnes aasta 2006 oktoobris. Lisaks hüljatusele on rakenduse andmeid raske pildi kujul eksportida.

Tulenevalt rakenduse mahajäetusele, piiratusele Windows platvormile, skeemi pildi ning SQL päringute eksportimise keerukusele soovib autor võtta kasutusele uue andmebaasi mudelite visualiseerimist võimaldava rakenduse. Autor võrdles erinevaid olemasolevaid lahendusi, aga erinevate puudujääkide tõttu otsustas arendada enda lahenduse.

2.2 Lahendatav probleem

Andmebaasi skeemide visualiseerimiseks on palju erinevaid tööriistu. Tööriistad jaotuvad enamasti kaheks. Esimesed on tasuta/odavad üldiste skeemide koostamiseks loodud tööriistad ja teised on kallid andmebaasiskeemide loomiseks spetsialiseeritud tööriistad. Üldiste skeemide koostamist võimaldavad tööriistad keskenduvad rohkem stiili valikutele ja ei sisalda piisavalt funktsionaalsust. Spetsialiseeritud tööriistade kasutamine on piiratud tänu kallile hinnale ja problemaatilisele kasutajakogemusele.

Enamasti lubavad skeemide loomist võimaldavad rakendused genereerida SQL päringuid andmebaasi tabelite loomiseks. Tegu on kasuliku omadusega, mis ilmneb igas väärikas andmebaasi skeemide koostamisele spetsialiseerunud rakenduses, aga tänapäeval kasutatakse aina rohkem ORM raamistikke, mis asendavad SQL päringute vajaduse.

Sellest tulenevalt ei oma ER skeemi tarkvara kasutamine väljaspool andmebaasi skeemide dokumenteerimist ja olemasolevate andmebaaside kalla töötamist erilist tähtsust. Soovitatav rakendus peaks võimaldama andmebaasi skeemi põhjal luua mudeleid vastavalt kasutaja vajadusele. Hetkel puudub kasutajal võimalus tema vajadustele vastavat koodi läbi skripti kergesti genereerida. Sellist funktsionaalsust lubavad küll järgnevalt selle töö raames võrreldud MySQL Workbench ja DbSchema rakendused, aga analüüsi käigus osutus see funktsionaalsus halvasti toetatuks.

Populaarsed koodihoidlad Github, Gitlab ja Bitbucket võimaldavad projektide dokumenteerimist Markdown formaadis. Antud formaat lubab skeemi pildi kujul esitamist, aga skeemi uuenemisel muutub piltide haldamine probleemseks, sest uue pildi versiooni genereerimine nõuab ligipääsu pildi loomiseks kasutatud rakendusele ning selle rakenduse skeemifailidele ja pildi täpsed muudatused ei kajastu versioonihalduses. Selle asemel võiks kasutada skeemide kuvamiseks tekstikujulist formaati ja Markdown koodiblokk elemente, sest sellise projekti dok.-i (dokumentatsiooni) on lihtsam muuta ja tehtud muudatused kajastuvad versiooni ajaloos selgemini.

Loodav rakendus peaks olema tasuta kättesaadav ja töötama sõltumata kasutaja operatsioonisüsteemi platvormist. Skriptimise soodustamiseks peaks rakendus olema avatud lähtekoodiga.

2.3 Projekti eesmärk

Lõputöö eesmärgiks on arendada tasuta kättesaadav ja avatud lähtekoodiga andmebaasi skeemide visualiseerimiseks mõeldud veebirakendus. Lahendus erineb alternatiividest võimaluse poolest lubada kasutajal programmi väljundit enda loodud skriptidega kergesti laiendada ja põhimõttest esitada ning hoida andmeid tekstikujulisel formaadil. Loodav rakendus on lihtsasti kasutatav, sest selle arendamisel on peetud silmas alternatiivsete rakenduste kasutajaliidese ja kasutajakogemuse vigu.

Kuna rakendus ei suuda kõigi kasutajate vajadusi ise rahuldada, peab rakendus lubama kasutajal programmi vastavalt oma vajadustele läbi skriptimiskeele laiendada. Sellega seoses peab rakendus sisaldama võimalust skriptide laadimiseks, käivitamiseks, muutmiseks ja loomiseks. Skriptimine on mõeldud ennekõike programmi väljundi genereerimiseks. Erinevate vajaduste jaoks skriptide loomine on antud töö skoobist väljas

ja jääb programmi kasutaja ülesandeks. Skriptimise soodustamiseks on rakendus loodud avatud lähtekoodiga, et skriptide autorid oleksid õhutatud jagama loodud skripte tagasi rakenduse arendajatega ja tagada seeläbi mahukas ja kvaliteetne sisseehitatud skriptide kogum.

Rakendus esitab andmebaasi skeeme teksti formaadis. See laiendab rakenduse skeemide eksportimise võimalusi ja lubab skeeme lihtsasti muuta. Lisaks pildi formaadile saab rakenduse skeeme esitada teksti kujul veebilehekülgede koodiplokkide sees. Antud lahendus sobib populaarsete Github, Gitlab ja Bitbucket koodihoidlate võimalusega dokumenteerida projekte Markdown formaadis. Selline esitlusformaat lubab hiljem antud skeemi väiksemas mahus lihtsalt muuta ilma selle loomiseks kasutatud rakendust avamata. Lisaks on sellises formaadis skeemile tehtud muudatused versioonihalduses paremini jälgitavad.

3 Olemasolevate rakenduste võrdlus

Järgnevalt võrreldakse kaheksat erinevaid skeemi loomise rakendusi. Võrreldavad rakendused on valitud kajastama erinevate põhimõtete ja hindadega tooteid. Kõigepealt seatakse võrdluseks üldised kriteeriumid. Seejärel võrreldakse rakendusi ja lõpuks võetakse võrdlus kokku erinevate rakenduste puudustega ning koostatakse tabel võrreldavate rakenduste erinevuste ja sarnasuste kajastamiseks.

Pärast võrdlemist loetletakse autori tähelepanekuid põhinedes erinevate rakenduste kasutamise kogemusele. Antud tähelepanekud mõjutavad loodava rakenduse kasutajaliidese ja kasutajakogemuse osas langetatavaid otsuseid.

3.1 Võrdluse kriteeriumid

Rakenduste paremaks võrdluseks on defineeritud erinevad kriteeriumid, mille põhjal saab rakendusi lihtsamini kategoriseerida ja hinnata. Hindamisel võetakse arvesse rakenduse tööplatvormi, üksikisiku hinda, autori arvamust ja ekspordi funktsionaalsust. Lisaks tuuakse välja muud tarkvara spetsiifilisi omadusi ning võimeid, mis eelnevate kriteeriumite alla ei sobi. Kriteeriumid aitavad olemasolevaid rakendusi paremini võrrelda, aga ei kajasta loodava rakenduse funktsionaalsust. Antud kriteeriumite põhjal koostatakse võrdluse kokkuvõtte tabel.

3.2 Võrdluse sisu

Alguses võrreldakse QSEE SuperLite, sest selle puudused on lõputöö tegemise ajendiks. Edasi võrreldakse kahte populaarset veebipõhist skeemijoonistus tarkvara diagrams.net ja Lucidchart. Edasi võrreldakse kahte andmebaasidega töötamiseks mõeldud tarkvara MySQL Workbench ja DbSchema, mille funktsionaalsuse hulka kuulub skeemide koostamine. Edasi võrreldakse spetsiaalselt andmebaasi ER skeemide loomiseks mõeldud tarkvara Vertabelo. Lõpuks võrreldakse kahte skeemide joonistamiseks hoopis teistsugust lähenemist kastutavat QuickDBD ja Monodraw rakendust. Lisaks üritati võrrelda Erwin

tarkvara, aga sellele ebaõnnestus autoril litsents hankida. Lõpuks võetakse võrreldavad tooted kriteeriumite põhjal kokku.

3.2.1 QSEE SuperLite

QSEE SuperLite on Windows operatsioonisüsteemi tasuta rakendus erinevate skeemide koostamiseks. Rakenduse viimane versioon tuli välja aastal 2006 ja sellest tulenevalt näeb rakenduse kasutajaliides aegunud välja, aga sellele vaatamata sellele käivitub rakendus probleemideta. Rakendus võtab kõvakettal kõigest 4MB ruumi.

Tarkvara sisaldab lisaks skeemide koostamisele palju muud funktsionaalsust. Rakendus sisaldab funktsionaalsust skeemi koostamiseks olemasoleva andmebaasi põhjal, aga see nõuab ODBC draiveri paigaldamist ja autor sellega toime ei tulnud. Skeemi põhjal saab põhimõtteliselt koostada HTML dok.-i, aga genereeritud dok. on visuaalselt väga kehv, seega ei leiaks see ilmselt kuskil kasutust. Rakendus suudab skeeme valideerida ja skeemi põhjal SQL päringuid genereerida, aga antud funktsionaalsused on vigased. Valideerimine ei suuda korrektselt aru saada kahe omavahel seotud tabeli suhteväljade samaväärsete nimede erinevust skeemil ja SQL päringute genereerimine lisab tabelitele juurde skeemil olematuid välju, mis muudab antud võimekuse suuresti mõttetuks. Rakendus sisaldab nuppu PHP koodi genereerimiseks, aga nupp on näiliselt alati mitteaktiivne, seega ei saa PHP koodi genereerimist funktsionaalsuseks lugeda.

QSEE SuperLite skeemi saab eksportida XML ja PDF formaadis ning programmi olekut saab salvestada QSEE failiformaadis. Pildikujul skeemi saamiseks peab kasutaja PDF faili pildiks ümber konverteerima. XML formaat sisaldab programmi olekut ja seda saab kasutada programmi oleku taastamiseks, jääb selgusetuks QSEE failiformaadi ja XML failiformaadi erinevus rakenduse oleku salvestamiseks.

Eelnimetatud probleemide tõttu saab rakendust kasutada peamiselt vaid skeemide koostamiseks. Skeemide koostamiseks on rakendus rahuldav. Skeemid ei näita tabelite väljade väärtuse puutumatuse võimalust ja tabeli välja nimetus ning välja andmetüüp on kokku kirjutatud, mis vähendab loetavust. Tabelite ja nendevaheliste suhete joonistamine on kerge.

3.2.2 diagrams.net/draw.io

diagrams.net (varasemalt draw.io) on tasuta avatud lähtekoodiga rakendus erinevate skeemide joonistamiseks. Rakendust saab kasutada veebipõhiselt ilma konto loomiseta või võrguühenduseta Windows, Mac ja Linux operatsioonisüsteemidel. Rakendus kasutab korraga nime diagrams.net enamasti eraldiseisva tasuta rakenduse raames ja draw.io nime peamiselt tasulisel integratsioonil teiste toodetega, rakenduse failiformaadi ning rakenduse Github koodihoidla projektis. Sealhulgas kirjutatakse nimed väikese algustähe ja .io/.net domeeninime lõpuga [2].

Rakenduse kasutamine andmebaasi skeemide moodustamiseks on üsna keeruline. Andmebaasi skeemide loomiseks on rakenduses valmisolevad komponendid nagu tabelid, tabelite väljad ja tabelite vahelised suhtejooned, mida saab hiirega skeemile lohistada. Kahjuks on nendel komponentidel raske vahet teha, sest visuaalselt näevad nad samasugused välja ja puudub võimalus luua nimekirja kasutaja lemmikkomponentidest. Tabeli väljade loomine on ebaintuitiivne, selle saavutamiseks on kolm erinevat viisi. Tabeli välja valimine on üks problemaatilisemaid protsesse, sest rakendus ei saa aru kas valida tervet tabelit, välja või välja teksti. Puudub võimalus luua tabelit nime ja andmetüübi väljaga. Kasutajakogemus on halb.

Tabelite vaheliste suhtejoonte moodustamine on ebaloogiline, selleks on neli erinevat viisi ja nendest parim on tabeli peal hiirega hõljudes peaaegu nähtamatu risti sümboli hiirega kokku tõmbamine ühe teise tabeli ristisümboliga. Kasutajakogemus on halb, sest risti sümbolid on peaaegu nähtamatud ja kaovad tabeli peal vajutades ära. Tabelite vahelisi suhtejoonte omadusi saab kasutaja ise muuta ja jooni on võimalik käsitsi liigutada.

Skeeme saab eksportida PNG, JPEG, SVG, PDF ja HTML hetkeoleku pildina või iframe lingina, mida saab lisada veebilehekülgedele skeemi värskeima versiooni kuvamiseks. Skeemi saab jagada Google Drive ja OneDrive lingina, mis lubab erinevatel inimestel sama skeemi kallal korraga töötada ja kommentaare jätta.

3.2.3 Lucidchart

Lucidchart on veebipõhine programm skeemide joonistamiseks. Rakendus on tasuline, tasuta versioon on piiratud 3 skeemi ja 60 skeemi kujundini, antud piirang lubab ainult

väiksemaid andmebaase visualiseerida. Piirangu eemaldamine maksab 8€/kuu. Rakendus tuleb seitsmepäevase prooviperioodiga [3].

Lucidchart on andmebaasiskeemide koostamiseks päris võimas ja selle töövoog on intuitiivne. Rakendus sisaldab nelja valmis tabeli komponenti, mida saab hiirega skeemile lohistada. Tabelite suhete loomine toimub paigutatud tabelite küljest ja pärast joone paigutamist saab selle omadusi muuta. Erinevalt diagrams.net rakendusele valib Lucidchart tabeli välja peale vajutades selle välja ja väljade lisamine ning kustutamine on lihtne.

Lucidchart sisaldab peale andmebaasi skeemide joonistamisega seotud operatsioone. Skeemi põhjal on võimalik SQL päringuid eksportida. Lisaks annab rakendus kasutajale olemasoleva andmebaasi importimiseks SQL päringu, mille tulemuse üleslaadimisel koostab rakendus andmebaasi skeemi. Selle funktsionaalsuse demonstreerimiseks on valmis näidis andmebaasi skeemid.

Lucidchart lubab skeeme eksportida PDF, JPEG, PNG ja SVG pildina, iframe lingina ja Visio rakenduse failina. Lisaks saab importida andmeid Visio, Omnigraffle, Gliffy ja diagrams.net programmidest. Rakendus lubab mitmel erineval kasutajal ühe skeemi kallal korraga töötada.

3.2.4 MySQL Workbench

MySQL Workbench on Windows, Mac ja Linux operatsioonisüsteemi programm MySQL andmebaasidega töötamiseks. Rakenduse kogukonna versioon on avatud lähtekoodiga tasuta rakendus, mis sisaldab enamus standardversiooni funktsionaalsust, standardversiooni hinda mainitud ei ole. Tasuta versiooni ei ole võimalik dok.-i genereerimiseks ega skeemi valideerimiseks kasutada [4].

MySQL Workbench sobib ainult MySQL andmebaaside dokumenteerimiseks, aga rakendus erineb teistest oma professionaalsuse ja hinna suhtes. Rakendus pärineb 2003. aasta Michael G. Zinner arendatud avatud lähtekoodiga DBDesigner4 programmist [5].

Rakenduse allalaadimine Oracle leheküljelt on kasutajavaenulik, lehekülg proovib kasutajat panna Oracle kontot looma. Rakenduse saab ühendada andmebaasiga rakenduse käivitamise avalehelt ja ülariba menüüst, ülaribamenüü kaudu rakenduse andmebaasiga ühendamisel jookseb rakendus kokku. Rakendus sisaldab palju erinevat funktsionaalsust

nagu andmebaasi andmete otsimine, kuvamine, muutmine, importimine ja eksportimine, SQL päringute koostamine, Python skriptide loomine ja skeemide koostamine nullist või olemasoleva andmebaasi põhjal. Erinevad skriptide loomiseks mõeldud rakenduse osad on segased.

Rakendus sisaldab kuute lihtsat näidis skripti, aga need ei tööta seoses rakenduses kasutatud uuema Python prog.k.-e (programmeerimiskeele) versiooniga. Sisseehitatud tekstiredaktor annab soovida, sest koodi kopeerimisel tekkinud tühikud jäävad tekstiredaktoris märkamatuks ja põhjustavad skripti seiskumise. Skriptide koostamine väljaspool rakendust on keeruline dok.-i puudumise ja skeemi andmete JSON esitusformaadi puudumise tõttu.

Skeemi loomiseks antakse kõigest 4A suurune tööpind. Tegemist on segadust tekitava otsusega, Stack Overflow küsimus ja vastus leheküljel on skeemi joonistusala suuremaks muutmine häälte poolest populaarsuselt 5 probleem MySQL Workbench kasutamise kohta [6]. Skeeme saab luua visuaalselt, aga skeemil puudub asetatavate elementide eelvaade. Tabelite suhete visuaalne loomine ei tööta olemasolevate väljade ühendamisel, selleks peab tabeli suhteid käsitsi kirjutama. Tabeli valimine avab alumise seadistusriba, mille peab kogu aeg käsitsi sulgema, sest muidu muutub skeemi vaade väga väikeseks ja seega raskesti kasutatavaks. Tabeli väljade andmetüübid on kirjutatud välja nimedega kokku. Suhtejoonte paiknemist on peaaegu võimatu kontrollida ja paratamatult hakkavad jooned üle tabelite jooksma. Skeemi saab eksportida PNG, SVG ja PDF pildina ning erinevate SQL päringutena.

3.2.5 DbSchema

DbSchema on Java põhine Windows, Linux ja Mac operatsioonisüsteemidel töötav andmebaasi disainitöörist. Rakendus sisaldab palju erinevat funktsionaalsust, aga kahjuks ei ole see enamasti skeemide koostamisega seotud.

DbSchema skeemid on kollektsioon valitud tabelite paigutusest. See võimaldab korrektselt vormistatud tabelitest koostada kiiresti palju väikseid skeeme. DbSchema skeemid kuvavad väikseväärtusena tabeli väljade andmetüüpe krüptiliselt ühetäheliste sümbolitena, aga skeemi seadistuses on see lihtsasti muudetav. Suhtejooned on problemaatilised. Suhtejoonte koostamine ei ole loogiline protsess, suhtejooni on võimalik koostada sama tööriistaga kahel erineval viisil olenevalt rakenduse olekule enne

tööriista valimist. Joonte kustutamisel ei eemalda programm tabeli välja suhte sümbolit. Suhtejoonte ega skeemi enda muudatused ei allu võta tagasi käsule. Suhtejooni kasutaja ise paigutada ei saa, aga programm väldib joonte üle tabelite tõmbamist.

Skeemi saab eksportida PNG, GIF, JPG ja PDF pildina. Võimsamaks eksportimise võimaluseks on dok.-i genereerimine, mis sisaldab skeemi koos iga tabeli kirjeldusega. Dokumentatsiooni saab genereerida PDF, HTML ja MD formaadis, nendest 2 viimast sisaldavad skeemi SVG formaadis. SVG formaadi eelis on skeemi interaktiivsete elementide säilitamine, näiteks hõljudes hiirega tabeli välja peal kuvatakse välja andmetüüp, kommentaar ja olemasolul valgustatakse välja suhtejoon.

DbSchema on ideaalne eksisteerivate andmebaasidega töötamisel. DbSchema suudab eksisteeriva andmebaasi põhjal skeeme koostada, kuvada ja muuta erinevate tabelite väärtusi koos seotud relatiivsete väärtustega, koostada eksisteerivatele andmebaasidele SQL päringute migratsioone ja importida/eksportida tabelite andmeid CSV või XML failidest. DbSchema suudab luua skeemi tabelitele erinevaid SQL päringuid, mitte ainult tabeli loomise päringut. Lisaks sisaldab rakendus visuaalset päringu koostajat. See peaks SQL ühendpäringuid toetama, aga autoril neid moodustada ei õnnestunud. DbSchema sisaldab erinevaid võimalusi võlts andmete genereerimiseks, mis võib andmebaasi testimisel kasulikuks osutuda. Rakendus suudab käivitada Groovy skripte. Skripti käivitamisel kaovad rakenduse vea tõttu programmi vaatest suhtejooned ära. Skriptide koostamine ei sisalda tüüpiliste operatsioonide näidis skripte, sest skriptimisega on võimalik vältida rakenduse tasuta litsentsiga kaasnevaid funktsionaalseid piiranguid nagu skeemi dok.-i genereerimine.

Rakendus tuleb 14 päevase prooviperioodiga. Rakenduse tasuta versioon on väga piiratud funktsionaalsusega [7]. Rakenduse akadeemiline litsents maksab 98$, isiklik litsents 196$ ja ärilitsents 294$. Litsents on eluaegne, aga rakenduse uuendamiseks peab igal aastal maksma 50$ isikliku litsentsi eest ja 75$ ärilitsentsi eest [8].

3.2.6 Vertabelo

Vertabelo on võimas veebipõhine andmebaasi skeemide joonistamise tööriist. Vertabelo kasutamiseks peab looma konto. Vertabelo salvestab skeeme kasutaja konto alla ja lubab mitmel kasutajal korraga ühe skeemi kallal töötada.

Vertabelo kasutamine kiire skeemi moodustamiseks on tänu kasutajaliidese paigutusele üsnagi ajakulukas, näiteks ei toeta rakendus parema hiireklahvi vajutusega elemendi kontekstimenüü avamist, kasutaja peab keskenduma nii rakenduse kasutajaliidese keskel olevale skeemile, ülemisele tööribale ja vasakpoolsele tabeli seadistamise ribale. Tabelite vaheliste suhtejoonte paigutust saab muuta vaid vähesel määral ja suuremates skeemides hakkavad jooned tabelitest läbi minema. Skeemist on lihtne aru saada, tabelite väljad sisaldavad põhjalikke andmetüüpide kirjeldusi, mis hoitakse välja nimest eraldatult.

Vertabelo toetab eksporti SQL päringutena, PDF, HTML ja DOCX dokumentatsioonina ning PDF, SVG ja PNG pildina. Kahjuks on genereeritud dokumentatsioon sisult kehv. Lisaks on võimalus eksportida skeemi XML formaadis ja importida XML faile ning SQL päringuid, aga see funktsionaalsus asub rakenduse kasutajaliideses hoopis teises kohas. XML formaadi mõte jääb arusaamatuks, sest originaalset skeemi paigutust sellest taastada ei ole võimalik.

Skeemid kasutavad versiooni haldamist ja lubavad erinevate kasutajate koostööd. Skeemi andmed toetavad valideerimist vigade ja hoiatuste näol. Vertabelo suudab brauseris näidata SQL päringuid iga tabeli kohta, see annab tabeli väljadest hea ülevaate ja võimaldab üksikute tabelite SQL päringuid eksportida. Skeemide andmete haldamiseks on erinevad tabelipõhised kasutajaliidese formaadid.

Vertabelo tuleb 7 päevase prooviperioodiga. Kõige odavam 24$/kuu hinnatase, piirab kasutaja konto 20 skeemini ja iga skeemi 100 tabelini, järgmine hinnatase on mõeldud 5 liikmelisele meeskonnale ja maksab 142$/kuu, aga eemaldab eelnimetatud piirangud [9]. Ülikooli meiliga on võimalik toodet tasuta kasutada.

Rakenduse server lakkas kasutamise käigus vähemalt kaheks tunniks töötamast väljaspool ettenähtud hooldusaega veakoodiga 503.

3.2.7 QuickDBD

QuickDBD on veebipõhine ER skeemide koostamise lahendus. Antud lahendus on unikaalne selle suhtes, et skeemid koostatakse tekstina läbi veebipõhise tekstiredaktori. Selleks kasutatakse SQL keele sarnast lihtsustatud formaati.

Rakenduse kasutamise teeb intuitiivseks lihtsalt mõistetav kasutajaliides, näidisskeemi olemasolu ja programmi keele dok.-i. Tekstiredaktorisse sisestatud tekst töödeldakse ja

õnnestumise korral kuvatakse kasutajale visuaalne skeemi tulemuse formaat, vea korral kuvatakse veateade. Visuaalses formaadis saab korrigeerida tabelite paiknemist. Tabelite vahelisi suhteid saab luua ja kustutada visuaalselt kui ka tekstiredaktoris. QuickDBD laseb koostatud skeemi eksportida PNG ja SVG pildina, genereerida PDF ja RTF dokumentatsiooni ning koostada SQL päringuid. Programmi enda olekut saab salvestada ainult registreeritud kasutaja projekti lingina, antud lähenemine on mõistlik mitme kasutajavahelise koostöö lubamiseks. Miinusena tasub mainida, et tabelite vahelistest suhetest on raske aru saada ja suhtejooni ei saa käsitsi paigutada. Jooned on olemuselt SVG Bézier kõverad ehk sujuvad kõverjooned, neid võibolla tehniliselt lihtne kujutada, aga antud lahenduses on nad skeemil raskesti jälgitavad. QuickDBD sisaldab SQL failide põhjal skeemide loomist, aga see funktsionaalsus on veel beetaversioonis.

Antud lahendust saab proovida tasuta 1 skeemi ja 10 tabeli piires. Programmi normaalhind on 7$/nädal, 14$/kuu või 95$/aasta. Rakenduse kasutuse propageerimiseks on võimalik saada 2 kuud tasuta jagades programmi Twitter keskkonnas või 12 kuud tasuta kirjutades programmist 500 sõnalise blogi artikli [10].

3.2.8 Monodraw

Monodraw on Mac operatsioonisüsteemi ASCII kunsti redaktor. Rakendus lubab luua diagramme, vooskeeme, mõttekaarte, ER skeeme ja kunstilisi joonistusi. Rakendus on saadaval ühekordse 8.99€ ostuna. Rakendus tuleb 30 päevase prooviperioodiga [11].

Rakendus lubab luua täisnurkseid ristküllikuid, teemandikujulisi rombe ja ellipseid, mille sees ja külgedel saab kujutada erinevate vormistusvõimalustega teksti. Kujundeid saab omavahel joontega ühendada. Joonte algus- ja lõpp-punkte saab määrata erinevate sümbolite vahel, ER skeemide jaoks on olemas spetsiaalne varesejala tähistus formaat. Rakendus sisaldab külgriba olemasolevate kujundite grupeerimiseks, peitmiseks ja kihi haldamiseks. Rakenduse skeeme saab eksportida teksti kujul ja SVG ning PNG pildina.

Rakenduse kasutamine andmebaasi skeemide loomiseks on piiratud kontseptuaalsete skeemidega. Rakendus lubab tabelite vahelisi suhteid luua ainult ristkülliku külgede keskpunktidest, see piirab ühe olemi suhted nelja teise olemini. Rakenduses tekstikujul eksportimine on problemaatiline erinevate sümbolite korrektsel kuvamisel.

3.3 Võrdluse kokkuvõte

Võrreldavad rakendused on olemuselt erinevad, aga kõik lubavad koostada skeeme, mida saab kasutada andmebaaside visualiseerimiseks. Tabel 1 kirjeldab võrreldud rakendusi vastavalt kriteeriumitele. Võrdluse käigus tuvastati probleeme QSEE SuperLite rakenduse kasutamises, mille tulemusena langes rakenduse kasulikkus ainult skeemide joonistamisele.

Tabel 1. Olemasolevate rakenduste võrdluse tulemus.

Rakenduse nimi	**Tööplat-vorm**	**Hind**	**Autori arvamus**	**Eksport**	**Muud omadused**
QSEE Supelite	Windows	Tasuta	Skeemide loomine on lihtne. SQL päringute genereerimine on vigane. Skeem on halvasti loetav, sest tabeli välja väärtuse puudumise võimalus ei kajastu skeemil ja tabeli välja nimed ning andmetüübid on kokku kirjutatud. ODBC draiverit ja PHP koodi genereerimist kasutada ei õnnestunud.	Skeem (PDF)	Võtab kõvakettal ainult 4MB ruumi
diagrams.net	Veeb, Windows, Linux, Mac	Tasuta	Kasutajaliides on segane, samu asju on võimalik mitut erinevat viisi teha. Ei ole võimalik skeemil kujutada tabeli väljade andmetüüpe. Kasutamine on raske. Ei soovita!	Skeem (PNG, JPEG, SVG, PDF, HTML, iframe)	Tasuta koostöö lubamine ja kommentaaride jätmine läbi Google Drive ja OneDrive kontode
Lucidchart	Veeb	8 €/kuu	Väga lihtne kasutada. Saadud skeemid näevad ilusad välja.	Skeem (PNG, JPEG, SVG, PDF, iframe), Visio, SQL	Koostöö lubamine, import Visio, Omnigraffle, Gliffy ja diagrams.net rakendustest ning andmebaasi import skeemiks

Rakenduse nimi	**Tööplat-vorm**	**Hind**	**Autori arvamus**	**Eksport**	**Muud omadused**
MySQL Workbench	Windows, Linux, Mac	Tasuta	Loodavatel skeemi elementidel puudub eelvaade. Kasutajaliides on problemaatiline. Rakendus jookseb teatud oludes kokku. Skeemi välja nimed ja andmetüübid on kokku kirjutatud ja seega raskesti eristatavad. Skeemi tabelite automaatsed suhtejooned jooksevad üle teiste tabelite.	Skeem (PNG, SVG, PDF, SQL)	Andmebaasi andmete kuvamine, muutmine, eksportimine ja importimine, Python skriptide loomine ning andmebaasi importimine skeemiks
DbSchema	Windows, Linux, Mac	196 $ [1]	Relatiivsete andmete kuvamine on hea. Skeem on ilus, sest tabelite automaatsed suhtejooned väldivad üle tabelite jooksmist. Sobib rohkem andmete haldamiseks, skeemide koostamine on probleemne. Skeemi elemendid ei allu võta tagasi käsule. Suhtejoonte kustutamine ei kajastu skeemi tabeli suhte sümboli kaotamises. Skripti käivitamisel kaovad suhtejooned ekraanilt.	Skeem (PNG, GIF, JPEG, PDF, SVG), dok. (PDF, HTML, MD), SQL	Andmebaasi andmete kuvamine, muutmine, eksportimine, võlts andmete genereerimine, Groovy skriptide loomine ja andmebaasi importimine skeemiks
Vertabelo	Veeb	24 $/kuu	Kasutajaliides on väsitav, sest kasutaja peab keskenduma kolmele eri kohale. Puudub parema hiireklahvi kontekstimenüü. Rakenduse server lakkas töötamast väljaspool ettenähtud hooldusaega.	Skeem (PNG, SVG, PDF). dok. (PDF, HTML, DOCX), SQL	Koostöö lubamine, skeemi valideerimine, erinevad vaated skeemi haldamiseks, skeemi loomine SQL päringute põhjal
QuickDBD	Veeb	14 $/kuu	Skeemide koostamine on kiire, aga tabelite suhtejooni on raske jälgida. Hinna ja võimekuse aspektist on Lucidchart enamasti parem.	Skeem (PNG, SVG), dok. (PDF, RTF), SQL	Koostöö lubamine, skeemide koostamine läbi lihtsustatud SQL keele, SQL päringute import

[1] Rakenduse uuendamine igal järgneval aastal maksab 50$.

Rakenduse nimi	Tööplat-vorm	Hind	Autori arvamus	Eksport	Muud omadused
Monodraw	Mac	8.99 €	Kasutada on lihtne, aga andmebaasi skeemide koostamiseks üpriski piiratud. Skeemis kasutatakse Unicode sümboleid, mida ei suuda kõik rakendused korrektselt kuvada.	Skeem (PNG, SVG, TXT)	Tekstipõhise kunsti loomine

Skeemide joonistamiseks saab kasutada diagrams.net ja Lucidchart rakendust. Lucidchart on võimsam ja lihtsamini kasutatav kui diagrams.net, aga diagrams.net on tasuta ja töötab ka vajadusel ilma internetita. MySQL Workbench ja DbSchema sisaldavad küll palju erinevat funktsionaalsust andmebaasidega töötamiseks, sealhulgas skriptide loomise võimalust, aga skriptimine on esimeses rakenduses täiesti unustatud võimekus ja teises ei kajasta sisseehitatud näidisskriptid kogu skriptimise funktsionaalsust, sest see võimaldab ligipääsu tasuta litsentsile piiratud operatsioonidele. Mõlema rakenduse kasutuskogemus skeemi moodustamiseks on halb. Vertabelo ja QuickDBD lubavad mõlemad skeeme eksportida dok.-ina, aga samas on tegu veebipõhiste rakendustega, mis võivad igal ajal kättesaamatuks muutuda. Monodraw võimaldab luua vaid primitiivseid skeeme. Võrredud rakenduste skriptimise ja teksti kujul skeemi esitamise võimalustega autor rahule ei jäänud, lisaks oli paljude rakenduste kasutamine problemaatiline.

Skeemide koostamiseks on kõige parem olemasolev rakendus Lucidchart.

3.4 Võrdluse tähelepanekud

Järgnevalt nimetatakse võrdluse käigus tehtud autori tähelepanekuid, mida lõputöö raames jälgida. Tegemist ei ole võrdluse kokkuvõttega, sest osa nimetatavatest punktidest on triviaalsed eraldi mainimise pälvimiseks üksikute tarkvarade kirjelduses, aga paistavad silma erinevate rakenduste kasutamiskogemuse olemasolul. Koos tähelepanekuga nimetatakse rakenduse nimed, mille vigade põhjal selline järeldus tehti. Antud tähelepanekud aitavad tagada loodavale rakendusele parema kasutajaliidese ja kasutajakogemuse:

- Skeemi tabelite välja nime ja andmetüübi eraldatult hoidmine tõstab skeemi loetavust (QSEE SuperLite, MySQL Workbench)

- Sama tulemuse saavutamine erinevatest kasutajaliidese elementidest tekitab kasutajates segadust (suhtejoonte loomine rakendustes diagrams.net, MySQL Workbench, DbSchema)
- Hoides kasutajaliidest minimaalsena kergendab rakenduse kasutamist (diagrams.net)
- Skeemi tabelite muutmiseks on kõige intuitiivsem kuvada rakenduse keskel *modal* akent, see tagab lihtsama töövoo sujumise (MySQL Workbench, Vertabelo)
- Sirgjoonelised suhtejooned on loetavamad kui sujuvad kõverjooned (QuickDBD)
- Automaatselt genereeritud jooned peaksid vältima teiste joontega ühtimist ja üle tabelite jooksmist (MySQL Workbench, Vertabelo, QuickDBD)
- Graafiline joonte loomine ilma tagasisideta on kasutajavaenulik (MySQL Workbench, DbSchema)
- Mitteaktiivsete kasutajaliidese elemendid peaksid nende peal hiirega hõljudes kuvama mitteaktiivsuse põhjust, vastasel juhul ei oska kasutaja neid kasutada (QSEE SuperLite, Vertabelo)
- Tekstiformaadis jooniste eksportimisel tuleks vältida Unicode kasti joonistus sümboleid näiteks U+2571, sest nende kuvamise on problemaatiline (Monodraw)
- Rakendustes kasutatav miniakna element on skeemi navigeerimiseks üpriski kasulik, aga võib samas võtta kasutajaliideses liigselt ruumi ja seega on selle kasutuselevõtmine loodavas rakenduses küsitletav (MySQL Workbench, DbSchema, Vertabelo)
- Veebipõhiste lahenduste server võib igal hetkel alla anda (Vertabelo)

4 Loodava rakenduse analüüs

Järgnevalt analüüsitakse rakenduse nõudeid ja põhjendatakse erinevate tehnoloogiate sobivust neid nõudeid rahuldada. Analüüsi käigus selgitatakse välja sobiv prog.k. ja raamistik. Pärast seda analüüsitakse rakenduse arenduses kasutatavaid teeke.

4.1 Nõuded rakendusele

Rakenduse nõuded on määratud tulenevalt rakenduse iseloomust, olemasolevatest alternatiivrakendustest ja autori soovidest. Järgnevalt loetletakse arendatava rakenduse lõputöö raames oleva skoobi üldiseid nõudeid ja rakenduse erinevate olekute funktsionaalseid nõudeid. Hilisema edasiarenduse võimalusi käsitletakse lõputöö viimases peatükis enne kokkuvõtet.

4.1.1 Üldised nõuded

Järgnevalt kirjeldatakse rakenduse üldiseid nõudeid.

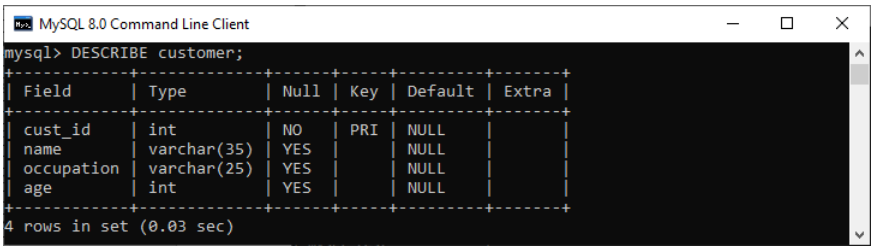
Programm peab sisaldama skriptimiskeelt. Programmi võimekuse hulka kuulub kasutajate loodud skriptide käivitamine. Skriptimine lubab kasutajal rakenduse väljundit vastavalt enda vajadustele muuta. Programm peab laadima ja käivitama sisseehitatud skripte ning kasutaja peab saama koostada uusi skripte ja muuta olemasolevaid. Skriptimise võimaldamiseks peab rakendus sisaldama skriptimiskeele teeki või peab rakenduse arendamiseks valitud prog.k. sisaldama skriptimise võimekust sisseehitatult.

Programm on tasuta ja avatud lähtekoodiga. Rakenduse skriptimise võimekuse toetamiseks peab rakendus olema tasuta ja avatud lähtekoodiga, et rakenduse kasutajad oleksid huvitatud rakenduse sisseehitatud skriptide kogumikku laiendada enda skriptidega.

Programm töötab sõltumata platvormist. Programm ei tohi olla piiratud kindlale operatsioonisüsteemile. Selle vältimiseks on rakendus loodud veebitehnoloogiatega, mis tagavad rakenduse töö läbi veebi brauseri sõltumata kasutaja operatsioonisüsteemist.

Programmi loomiseks kasutatav tehnoloogia on optimaalse võimekusega. Programmi jõudlust tuvastatakse kasutades ekraani kaadrite värskendussagedust sekundi jooksul muutuva graafika kuvamisel test rakenduses. Testkeskkonnaks on Windows 10 operatsioonisüsteem, Intel Core i5-4690K protsessor, 16GB mälu, Nvidia GeForce GTX 1060 6GB videokaart ja rakenduse käivitamiseks kasutatakse Mozilla Firefox ning Google Chrome brausereid. Antud lähenemine ei ole ideaalne, sest testkeskkond on valitud autori kättesaadavuse põhjal ja testid võivad olla ebaoptimaalselt kirjutatud, aga lasevad erineid tehnoloogiad võrrelda sarnaselt nagu neid oleks kasutatud päriselt rakenduses arenduseks.

Programm kasutab skeemide kuvamiseks ja salvestamiseks tekstipõhist formaati. Rakenduse skeemi salvestamise ja kuvamise formaat peab olema sama, et andmebaasi skeemi joonistusi saaks jagada ühtse formaadina, mida rakendus oskab importida ilma vahepealsete andmeformaatide kasutamiseta. Stiililt on antud projektis kasutatav skeemi formaat inspireeritud MySQL andmebaasi käsurealt kasutava kuju formaadiga (Joonis 1).



Joonis 1. MySQL andmebaasi tabeli käsurealt kuvamise formaat [12].

Teksti kujul andmeid saab läbi tekstiredaktori väljaspool rakendust piiratud määral kergesti muuta. See lihtsustab väiksemate muudatuste tegemise ilma vajaduseta omada ligipääsu rakendusele. Lisaks lubab antud formaat rakenduse salvestust Markdown failidesse lisada, mida erinevad populaarsed koodihoidlad (Github, Gitlab, Bitbucket) suudavad HTML lehekülgedena kuvada. See lubab dokumenteerida projektide andmebaasi skeeme otse nende projektide koodihoidla lehekülgedel ja antud skeemi muudatused on hästi versioonihalduses kajastatud.

4.1.2 Funktsionaalsed nõuded

Järgnevalt fikseeritakse loodava rakenduse funktsionaalsed nõuded. Rakendusel puudub võimalus kasutajate loomiseks, seega ei saa nõudeid erinevatesse kasutajagruppidesse

jaotada, selle asemel on nõuded grupeeritud vastavalt rakenduse kasutajaliidese olekule (Tabel 2).

Tabel 2. Rakenduse funktsionaalsed nõuded.

Oleku nimi	**Funktsionaalsus**
Index	Pealehekülg rakenduse tutvustamiseks, kasutaja suunatakse edasi Draw vaatesse
Draw	Skeemi vaate liigutamine
Draw	Skeemi vaate suumimine
Draw	Skeemi tabelite valimine
Draw	Skeemi tabelite lisamine
Table	Skeemi tabelite kustutamine
Draw	Tabeli asukoha muutmine skeemil
Table	Tabeli ridade lisamine
Table	Tabeli ridade kustutamine
Table	Tabeli rea nime välja muutmine
Table	Tabeli rea andmetüübi välja muutmine
Table	Tabeli rea atribuudi välja muutmine
Table	Tabeli ridade üksteise suhtes ülesse ja alla liigutamine
Table	Tabelite vaheliste suhtejoonte loomine
Draw	Skeemi tekstifailina salvestamine
Draw	Skeemi PNG pildina salvestamine
Draw	Skeemi tekstifailist laadimine
Script	Skeemi andmete JSON kujul kuvamine
Script	Skriptide loomine läbi tekstiredaktori
Script	Skriptifailidele nime panemine
Script	Olemasolevate skriptide sisu kuvamine
Script	Skripti käivitamine
Script	Sisseehitatud skript SQL CREATE päringute koostamiseks skriptimise demonstreerimise eesmärgil

4.2 Tehnoloogia analüüs I

Järgnevalt kirjeldatakse võimalusi rakenduse arenduses kasutada erinevaid raamistikke ja programmeerimiskeeli. Kõigepealt määratakse analüüsi kriteeriumid, et erinevaid

raamistikke paremini omavahel võrrelda. Edasi selgitatakse raamistike kasutamise tausta ja tuuakse välja võrreldavad raamistikud. Siis tutvustatakse igat raamistikku põhjalikumalt ja kirjeldatakse raamistiku vastavust kriteeriumitele. Lõpuks tehakse raamistike analüüsi kokkuvõte ja tehakse valik rakenduses kasutatava raamistiku kohta. Rakenduses kasutatud prog.k. oleneb valituks osutunud raamistiku põhjal.

4.2.1 Analüüsi kriteeriumid

Raamistikke võrreldakse 4 aspektist: skriptimise võimalus, dok.-i kvaliteet, populaarsus ning jõudlus. Aspekte hinnatakse punkti skaalal kasutades järgnevas tabelis väljatoodud värviskeemi (Tabel 3).

Tabel 3. Analüüsi kriteeriumid.

Punktide arv	**Skriptimise võimalus**	**Raamistiku dok.**	**Raamistiku populaarsus**	**Jõudlus testrakenduses**
0 – puudulik	Ei leidu ühtegi skriptimiskeelt.	Raamistiku paigaldamiseks on juhend, aga raamistiku enda kasutamine on arusaamatu.	Github tähti = [0, 2500)	Rakendus töötab jõudluse tõttu väga halvasti, rakendus jookseb kokku, FPS = (0, 10].
1 - halb	Leidub ainult 1 skriptimiskeel.	Raamistiku kasutamiseks on küsitletava kvaliteediga juhend, mis sisaldab aegunud infot või katkisi viiteid.	Github tähti = [2500, 5000)	Rakendus kasutajaliides on tihti mittereageeriv, FPS = (10, 30].
2 – rahuldav	Leidub üks üle 1000 Github tähega skriptimiskeel.	Raamistiku kasutamiseks on põhjalik juhend, aga mõned raamistiku aspektid jäävad selgusetuks.	Github tähti = [5000, 10000)	Rakendus ei võimalda jõudluse pärast hõljuv eelvaadete loomist, FPS = (30, 60].
3 – hea	Raamistiku prog.k. toetab skriptide interpreteerimist või leidub mitu üle 1000 Github tähega skriptimiskeelt.	Raamistiku kasutamiseks on põhjalik ja kvaliteetne juhend, mis sisaldab suures koguses koodijuppe ning kvaliteetset API dok.-i.	Github tähti = [10000, ∞)	Rakendus töötab kiiresti ja selle kasutajaliides on reageeriv, rakenduse tehnoloogia valik ei mõjuta rakenduse disaini otsuseid, FPS = (60, ∞).

Bakalaureusetöö mahu ja autori aja piiratust arvestades on skriptimise võimalust võrreldud üldiselt vastavalt selle raamistiku prog.k.-ele. Jõudluses kirjeldatakse selle raamistikuga tehtud testrakenduse graafika kuvamise sagedust kaadrit sekundis (*FPS*). Raamistiku populaarsust on hinnatud Github tähtede põhjal, andmed võetud 2022. September.

4.2.2 Raamistiku kasutuselevõtmise taust

Võib väita, et veebirakenduse arendamiseks ei ole raamistikku tarvis, piisab vaid JavaScript prog.k.-est rakenduse interaktiivsuse tagamiseks ja HTML elementidest kasutajaliidese loomiseks. Igal raamistikul on omad plussid ja miinused, mis aitavad või aeglustavad rakenduse arendust, aga raamistik ei ole kohustuslik osa veebirakenduse arendusest.

See väide võib teatud veebirakenduste puhul tõeks osutuda, aga antud töö raames loodav rakendus sisaldab palju kiiresti muutuvaid kasutajaliidese elemente, mille kuvamisega ei tule rakendus niisama toime. Vastavalt võimekuse nõudele tehti rakenduse jõudluse test, mille tulemuseks saadi aeglaselt töötav rakendus. Rakendus töötas Google Chrome brauseris 21 FPS ja Mozilla Firefox brauseris 9 FPS kaadrisagedusel [13].

Jõudluse tõstmiseks prooviti kolme erinevat ideed:

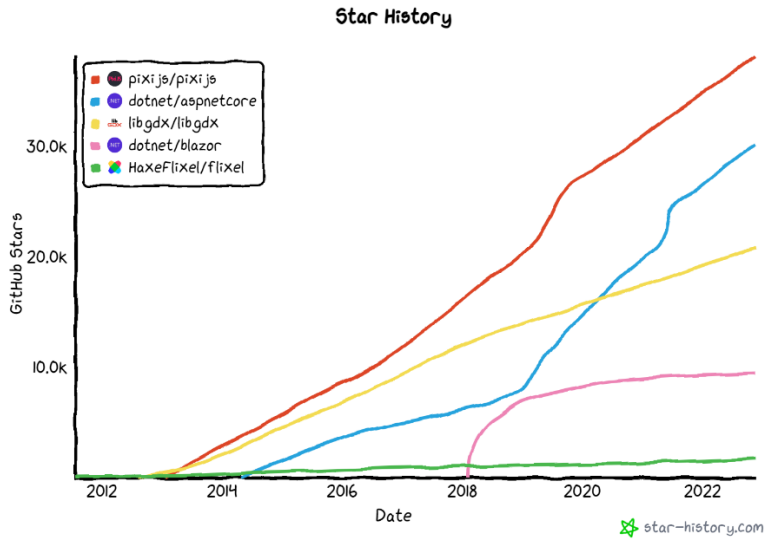
- Rakenduse jõudluse tõstmiseks saaks kasutada HTML *canvas* elementi, mis on loodud madalatasemeliseks kihiks dünaamilise graafika kuvamiseks. Elemendi otse kasutamine on selle madalatasemelisest olemusest tingituna keeruline, aga vahepealne graafiline raamistik peaks selle probleemi lahendama. Üheks selliseks graafiliseks raamistikuks on PixiJS.
- Teiseks võimaluseks on brauseris kasutatud JavaScript prog.k.-est tulenev aeglus. Selle vältimiseks saab vahetada aeglase JavaScript prog.k.-le kiirema WebAssembly prog.k.-le vastu välja. Selle teostamiseks saab kasutada Blazor raamistikku.
- Kolmandaks võimaluseks on kasutada graafika kuvamiseks mänguraamistikku. Töö käigus võrreldakse libGDX ja HaxeFlixel mänguraamistikku.

Vaatamata lähenemisele, osutub raamistiku kasutamine rakenduse arenduseks vajalikuks. Edasi kirjeldatakse võrreldavaid raamistike täpsemalt.

4.2.3 Raamistike ülevaatlik võrdlus

Raamistikke on vaja arenduse kiirendamiseks, rakenduse töötamiseks erinevatel platvormidel, rakenduse jõudluse tagamiseks, sisendi saamiseks, kasutajaliidese kuvamiseks ja teistest elementaarsetest vajadustest. Omavahel võrreldakse PixiJS, Blazor, HaxeFlixel ja libGDX raamistikke.

Antud raamistike valikul on lähtutud erinevatest aspektidest. Tegu on näiliselt aktuaalsete raamistikega. Kõik väljatoodud raamistikud on aktiivselt korrashoitud, kõikide projektide koodihoidlatesse on viimase kuu jooksul (2022. Aug.) muudatusi tehtud. Tegu on täismahus tasuta kasutatavate ja avatud lähtekoodi raamistikega. Kõik loetletud raamistikud on Github koodihoidlast tasuta kättesaadavad. Järgnevalt on välja toodud raamistike populaarsus Github tähtede põhjal (Joonis 2).



Joonis 2. Võrreldud raamistike populaarsus Github tähtede põhjal [14].

Tabel 4 annab ülevaate valitud raamistike prog.k.-est ja raamistiku populaarsusest. Lisaks võrreldaks raamistike prog.k.-te sobivust võimaldada skriptimise funktsionaalsust loodavale rakendusele Google otsingu tulemusel leitud teekide näol.

Tabel 4. Võrreldud raamistike ülevaatlik võrdlus.

Raamistik	**Raamistiku populaarsus**	**Prog.k.**	**Prog.k. skriptimise võimalus**
PixiJS	37400 Github tähte [15].	JavaScript/ TypeScript	Sisseehitatud keele konstruktsioonid koodi interpreteerimiseks. Lahendus sobib hästi kasutaja vaatenurgast JavaScripti lihtsuse ja populaarsuse poolest.
Blazor	Originaalne arhiveeritud Blazor projekt omab 9400 Github tähte [16]. Migreerunud Blazor omab ASP.NET Core projekti nime all 30000 Github tähte [17].	C#	Sisaldab palju erinevaid võimalusi skriptimiseks: CS-Script, Dotnet-Script, Nlua, Jint, Scriban, Python.NET, IronPython, Roslyn avatud lähtekoodiga kompilaatori skriptimise moodul. Kõik nimetatud teegid sisaldavad vähemalt 1000 Github tähte (2022. Sept.).
HaxeFlixel	1630 Github tähte [18].	Haxe	Parimaks valikuks on Nicolas Cannasse loodud hscript. hscript ei sisalda mingisugust juhendit ega eraldi lehekülge. Selle kasutamiseks on kaks koodijuppi projekti Github leheküljel. Autor sattus segadusse hscript sõltuvuste haldamise kohta ja küsis abi Cannasse käest, aga rahuldavat vastust ei saanud [19]. hscript on ebapopulaarne lahendus, projekti Github koodihoidla sisaldab ainult 211 tähte [20].
libGDX	20500 Github tähte [21].	Java	Palju erinevaid teeke nagu näiteks Jython, Beanshell, LuaJ, NetRexx, Rhino, JRuby, Groovy. Rhino, JRuby ja Groovy on nimetatud teekides kõige populaarsemad, nad omavad Github koodihoidlas vähemalt 3000 tähte.

Kokkuvõtteks on kõikide raamistikega kaasnenud prog.k. võimalik skriptimist teostada. Kõik prog.k.-ed sobivad skriptimiseks hästi, välja arvatud Haxe, mis sisaldab ainult ühte segast ja madala populaarsusega skriptimise teeki. Populaarsuse aspektist on HaxeFlixel enamus inimeste jaoks teadmatu raamistik, kõik ülejäänud raamistikud on populaarsed.

4.2.4 Raamistike põhjalik võrdlus

Järgnevalt kirjeldatakse võrreldavaid raamistikke ja tehakse nende praktilise kasutuse võrdlus. Praktiline võrdlus hõlmab raamistiku dok.-i kvaliteeti ja jõudlust testrakenduses.

PixiJS raamistik

Antud lahendus kasutab JavaScript prog.k.-t rakenduse arendamiseks koos PixiJS raamistikuga jõudluse tõstmiseks. slay_lines organisatsiooni poolt koostatud graafika kuvamise raamistike jõudlustesti põhjal on tegemist parima JavaScript prog.k. graafika kuvamise raamistikuga [22].

PixiJS on JavaScriptil põhinev 2D graafika kuvamismootor, mis prioritiseerib kiirust. PixiJS kasutab sisimas madalatasemelist WebGL JavaScript API kihti graafika joonistamiseks, pakkudes kasutajatele võimsat ja lihtsat viisi WebGL kihiga suhtlemiseks. Peale graafika joonistamist pakub PixiJS puudutus sündmuste käsitlemist. See on teinud PixiJS populaarseks tööriistaks interaktiivsete graafikate ja mängude loomisel. PixiJS peab ennast aegunud Flash platvormi võimsaks, iseseisvaks ja moodsaks asenduseks. PixiJS rakendused töötavad hästi koos Cordova ja Electron raamistikega, lubades levitada rakendusi lisaks veebile ka mobiili ja lauaarvuti platvormidele [23].

PixiJS soovitab algajatele ühte kasutaja kittykatattack loodud juhendit ja ühte ametlikku juhendit. Mõlemad juhendid annavad raamistikust hea ülevaate. PixiJS sisaldab üle 100 näite, mis demonstreerib erinevat funktsionaalsust. Näited töötavad otse kliendi brauseris ja toetavad koodi reaalajas muutmist. Lisaks saab käivitada näiteid erinevate PixiJS versioonides [24]. PixiJS API sisaldab otsingu funktsionaalsust, klassi sisukorda, viitavaid muutuja tüübi dok.-ile ja põhjalikke kirjeldusi [25]. Kokkuvõttes on dok. väga hea.

PixiJS kasutuselevõtmine tõstis rakenduse võimekust võrreldes raamistiku mittekasutamisega vähemalt kolm korda. Jõudluse testimiseks koostatud testi tulemusena töötas testrakendus Google Chrome brauseris 62 FPS ja Mozilla Firefox brauseris 28 FPS kaadrisagedusel [13]. PixiJS võimaldab teksti kuvamiseks kasutada vektorfonte läbi *PIXI.Text* klassi ja rasterfonte läbi *PIXI.BitmapText* klassi. Rasterfontiga muutuva teksti kuvamine osutus umbes 20 korda kiiremaks kui alternatiivse TrueType formaadis fondi kasutamine.

Rasterfondina kasutab PixiJS AngelCode fondi formaati. AngelCode formaati saab genereerida kasutades AngelCode kasutaja poolt loodud BMFont tasuta Windows programmi. Antud programm võtab sisendiks TrueType fondi ja genereerib vastavalt parameetritele PNG pildi koos fondifailiga. Alternatiivsete programmidena tuuakse

AngelCode leheküljel välja Littera, Glyph Designer, ShoeBox ja Hiero [26]. Autori arvates on parimaks programmiks BMFont, häda korral võib kasutada ka Hiero programmi.

Kokkuvõttes on tegu suhteliselt kiire raamistikuga, mis on populaarne ja hästi dokumenteeritud. Lõputöö rakenduse arenduseks sobiks see hästi.

Blazor raamistik

Antud lahendus kasutab Blazor raamistikku ja C# prog.k.-t.

Blazor on moodne Microsofti veebiraamistik, mis kuulub suuremasse ASP.NET Core ökosüsteemi. Blazor pärib ASP.NET ökosüsteemi Razor Pages mallimootori ja MVC koodimustri. Blazor lisab veebirakendusetele interaktiivsust, mis lubab muuta lehekülgede sisu ilma tervet lehekülge uuendamata. Blazor koosneb mitmest mudelist: Blazor Server, Blazor WebAssembly, Blazor Hybrid, Blazor PWA ja Blazor Native. Antud lahendus kasutab Blazor WebAssembly mudelit. Blazor WebAssembly rakendus saadab kliendi brauserisse WebAssembly koodiks kompileeritud .Net interpretaatori koos terve rakenduse serveriga. See tähendab, et rakendus töötab pärast esmast laadimist täielikult kliendi brauseris, aga algne rakenduse laadimine on aeglane [27].

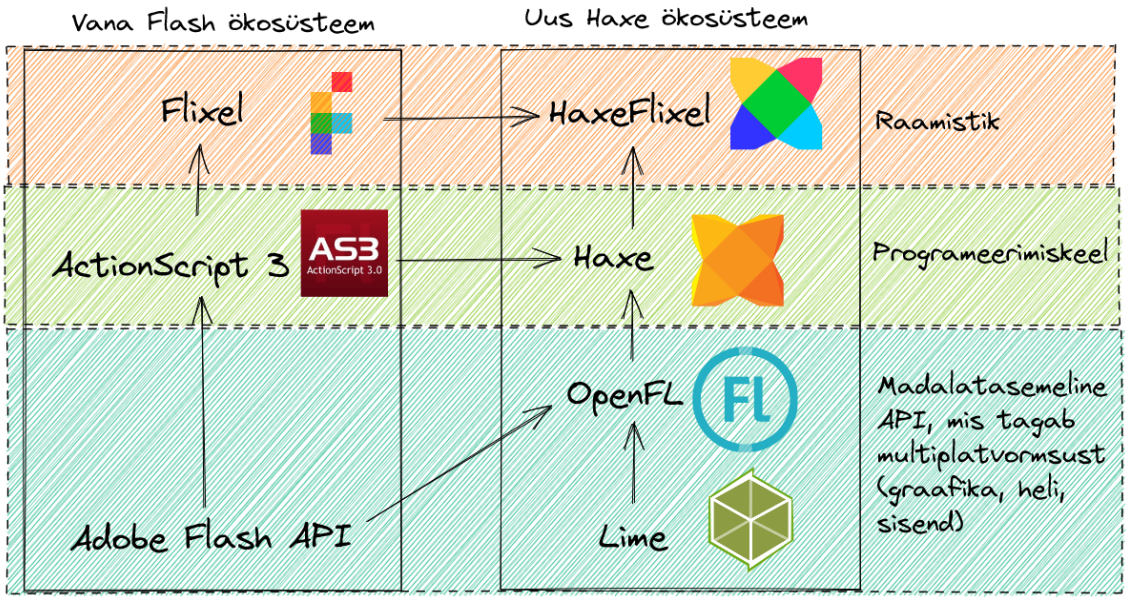
Microsofti ASP.NET Core dok. annab põhjaliku ülevaate Blazor tehnoloogiast. See sisaldab tehnoloogia lühitutvustust, erinevate Blazor moodulite kirjeldust ja kasutusjuhtumeid, projekti failide ja kaustade struktuuri selgitust, Razor mallimootori komponentide tutvustust ning lõpuks viiakse lugeja läbi koodijuppide kurssi Blazor spetsiifiliste komponentidega. Dokumentatsioon sisaldab ühte lühikest juhendit *TodoList* näidisrakenduse loomisest, aga selles ei kasutata ühtegi Blazor WebAssembly spetsiifilist omadust nagu näiteks JavaScript funktsiooni kutsumist C# funktsioonist või vastupidi. Juhend kehtib rohkem Razor Pages rakenduse kohta. Kahjuks on raske eristada Blazor Server ja Blazor WebAssembly jaoks mõeldud juhendeid. Puuduvad näited Blazor WebAssembly võimekuse demonstreerimiseks. Dokumentatsioon on oma stiililt mõeldud inimestele, kes on juba Blazor/ASP.NET Core rakendustega natuke tuttavad või kellele meeldib ohtralt dok.-i lugeda. Uue projekti loomine toimub läbi sisseehitatud malli, projekt tuleb kahe Blazor raamistiku võimekust demonstreeriva näitega [28]. Kokkuvõttes võib dok.-iga rahule jääda.

Blazor jõudluse testimiseks koostatud testrakenduses saavutati Google Chrome brauseris 15 FPS ja Mozilla Firefox brauseris 20 FPS kaadrisagedus. Lisaks selgus, et Blazor on küll suuteline kutsuma C# prog.k.-est JavaScript prog.k.-e funktsioone ja seega saab teoreetiliselt kasutada PixiJS raamistikku jõudluse tõstmiseks, aga kahjuks kahe erineva keele vahel andmete saatmine aeglustab programmi tööd [29].

Kokkuvõttes sisaldab Blazor arendajale mugavat C# prog.k.-t ja andmete sidumist Razor Pages mallimootorigi, aga JavaScript prog.k. kasutamine muutub raskemaks. Antud rakenduse arenduseks tarvis jõudlust Blazor raamistik ei võimalda.

HaxeFlixel raamistik

HaxeFlixel on avatud lähtekoodiga mitmeplatvormne 2D mänguraamistik, mis kasutab programmeerimiseks Haxe keelt ja on ehitatud OpenFL (Open Flash Library) teegi peale (Joonis 3). OpenFL on Flash ökosüsteemi jäljendus ja edasiarendus Haxe prog.k.-es. OpenFL on omakorda ehitatud Lime teegi peale, mis pakub süsteemitasemelist vundamenti mitmeplatvormsuse tagamiseks. HaxeFlixel on Flixel mänguraamistiku edasiarendus Haxe prog.k.-de [30].



Joonis 3. HaxeFlixel tehnoloogia päritolu ja sõltuvuse kirjeldus (autori Excalidraw joonis) [30].

HaxeFlixel kasutab Haxe prog.k., mille lõi Nicolas Cannasse 2005, et asendada Motion Twin mänguarendusettevõttes MTASC (Motion-Twin ActionScript 2 Compiler) avatud lähtekoodiga kompilaatorit. Algselt töötasid Haxe programmid Cannasse enda loodud Neko virtuaalmasinas ja AVM (ActionScript Virtual Machine) keskkonnas. Haxe hakkas

peagi toetama võimekust kompileerida teistesse keeltesse [31]. Tänapäeval on põhilisteks kompileerimissihtmärkideks Haxe enda interpretaator, JavaScript, HashLink, JVM (Java Virtual Machine) ja PHP7 (PHP: Hypertext Preprocessor) [32].

HaxeFlixel tehnoloogiast on keeruline aru saada ja HaxeFlixel dok. ei aita sellele kaasa. HaxeFlixel juhend kirjeldab HaxeFlixel paigaldamist, aga Lime teegist on juttu minimaalselt. Lime õppimiseks soovitatakse minna Lime https://lime.software/ leheküljele, aga see lehekülg on aegunud, uus lehekülg on https://lime.openfl.org/ [33]. Autorile jääb arusaamatuks, mida Lime täpselt teeb, seda kasutatakse käsurealt projekti ehitamiseks nagu kompilaatorit, aga erinevad allikad ei kirjelda seda kui kompilaatorit. Dok. sisaldab juhendit HaxeFlixel mängu loomiseks ja suurel hulgal erinevaid näited, mis töötavad kliendi brauseris [34]. API dok. on kohutav. Puudub struktuur, dokumenteeritakse peale HaxeFlixel enda veel palju kõrvalisi tööriistu, paljud funktsioonid on ilma kirjelduseta ja lisaks kasutatakse HaxeFlixel nime asemel Flixel nime. Ilma otsinguribata oleks API dok. täiesti kasutamatu [35].

Testi käigus saadud tulemuste põhjal võib HaxeFlixel raamistiku jõudlusega rahule jääda. HaxeFlixel raamisiku jõudluse testimiseks koostatud testirakendus töötas Google Chrome brauseris 63 FPS ja Mozilla Firefox brauseris 33 FPS kaadrisagedusel [36].

Kokkuvõtteks jääb HaxeFlixel raamistiku kasutamine lõputöö rakenduse loomiseks erinevatest aspektidest soovida.

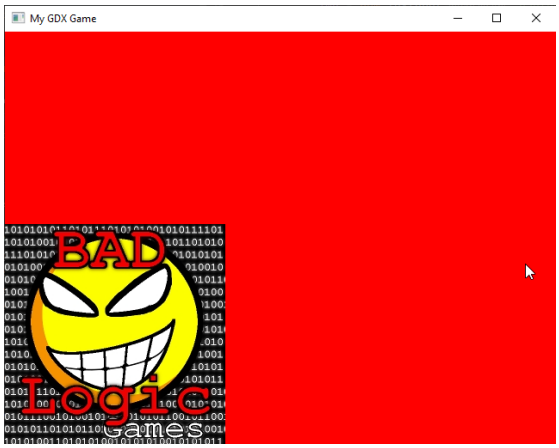
libGDX raamistik

Antud lahendus kasutab Java prog.k.-el põhinevat libGDX mänguraamistikku.

libGDX on Java prog.k.-es kirjutatud mitmeplatvormne avatud lähtekoodiga mänguraamistik. libGDX raamistiku lõi algelt arendaja Mario Zechner aastal 2009 Android platvormi põhiste mängude loomiseks. Algselt oli tegu Android platvormi põhilise raamistikuga, lauaarvuti tugi lisati peamiselt Android platvormi emuleerimiseks. See lubas koodi testimist lihtsustada ja seeläbi kogu arendusprotsessi kiirendada. 2012 võeti kasutusele GWT (Google Web Toolkit) tööriist, mis lubas libGDX rakendusel töötada veebi ja iOS platvormidel. Raamistik sai peagi mängu arendajate kogukonnas populaarseks, 2013 lõid vabatahtlikud sellele uue kodulehe. 2017 lõpetas Mario Zechner raamistiku edasise arenduse, aga tänu kogukonnale jätkus libGDX arendus, sellest ajast

saadik on jätkunud raamistiku uute versioonide avaldumine ja töö raamistiku kodulehe kallal [37].

libGDX raamistiku dok. kirjeldab erinevaid võimalusi raamistiku kasutamiseks ja raamistiku projekti struktuuri. Esmase projekti loomiseks kasutatakse eraldiseisvat Java prog.k. põhist libGDX Project Setup Tool (gdx-setup) rakendust [38]. Tööriistaga genereeritud projekt jätab ebaprofessionaalse mulje (Joonis 4).



Joonis 4. libGDX Project Setup Tool programmi loodud projekti kuvatõmmis.

Dok.-is väljatoodud näidisrakendused peaksid olema brauseris mängitavad, aga need viitavad https://libgdx.badlogicgames.com/demos/ leheküljele, mis ei tööta [39]. Selgus, et Bad Logic Games on arendaja Mario Zechneri kasutajanimi. See nimi esineb uuesti libGDX Project Setup Tool tööriista genereeritud projektis ja API moodulite nimetuses. Jääb arusaamatuks, miks seostatakse libGDX raamistikku aktiivselt Mario Zechneri kasutajaga, kui ta enam projekti arengus ei osale.

API dok. on väga suur ja otsingu funktsionaalsus puudub, funktsioone kirjeldatakse minimaalselt. Dokumentatsioon sisaldab 2 erinevat versiooni. Üks on üheleherakendus koos navigeerimise külgribaga ja teine on mitme leheküljeline rakendus ilma navigeerimise külgribata. Avades lingi dok. üheleherakenduse versioonis uues brauseri aknas minnakse üle dok. mitme leheküljelisele rakenduse versioonile. Selline erinevus tekitab segadust [40].

libGDX raamisik töötab hea jõudlusega. Testrakenduses töötas raamistik Google Chrome brauseris 120 FPS ja Mozilla Firefox brauseris 40 FPS sagedusega [41].

Kokkuvõttes on tegemist hea raamistikuga, aga juhendis esinenud katkiste linkide ja Bad Logic Games nime alleshoidmise pärast kahtleb autor raamistiku aktuaalsuses.

4.2.5 Analüüsi kokkuvõte ja tehnoloogia valik

Raamistike analüüsis käsitleti 4 erinevat prog.k.-t ja 4 erinevat raamistikku. Skriptimise poole pealt sisaldasid Java ja C# prog.k.-ed palju erinevaid väliseid teeke, JavaScript prog.k. sisaldab skriptimise funktsionaalsust sisseehitatult, soovida jäi ainult Haxe prog.k. skriptimise võimekus. Dokumentatsiooni poolest osutus näidete arvukuse poolest kõige paremaks PixiJS raamistik. Võrredud raamistikest osutusid populaarseteks PixiJS, Blazor ja libGDX. Jõudlusega tuli kõige paremini toime libGDX raamistik. Tabel 5 toob kokkuvõtvalt välja analüüsi tulemused.

Tabel 5. Tehnoloogia analüüsi kokkuvõte.

Raamistik ja prog.k.	**Skriptimiskeele olemasolu**	**Raamistiku dok.**	**Raamistiku populaarsus**	**Kaadrisagedus (Google Chrome; Firefox)**
Raamistikuta (JavaScript)	Javascript toetab skriptimist sisseehitatult	N/A	N/A	21;9
PixiJS (JavaScript)	Javascript toetab skriptimist sisseehitatult	Dokumentatsioon on hea, sisaldab palju näiteid.	37400 Github tähte	62;28
Blazor (C#, JavaScript)	Suurel hulgal valikuid väliste teekide näol.	Dokumentatsioon on rahuldav. Puuduvad näited Blazor võimekuse demonstreerimiseks - näidete põhjal on keeruline aru saada Blazor ja Razor Pages raamistike erinevusest.	30000 Github tähte	15;20
HaxeFlixel (Haxe)	Ainult segane ja vähe populaarne hscript skriptimise teek.	Dokumentatsioon sisaldab palju näiteid, aga on üldiselt siiski halb. API dok. on segane ja paljud funktsioonid on ilma kirjelduseta. Erinevate tehnoloogiate kasutuse eesmärk jääb segaseks.	1630 Github tähte	63;33
libGDX (Java)	Palju erinevaid populaarseid teeke.	Dokumentatsioon sisaldab vähe näiteid ja olemasolevad näited ei ole brauseris mängitavad 503 veateate tõttu, API dok. kasutamine on keeruline, Bad Logic Games nimi tekitab segadust.	20500 Github tähte	120;40

Analüüsi tulemusel osutuks parimaks lahenduseks PixiJS raamistik. Antud lahendus kasutab JavaScript prog.k.-t, mis sobib hästi skriptimise toetamiseks. PixiJS dok. osutus üheks parimaks. Võrreldavatest raamistikest osutus PixiJS kõige populaarsemateks.

Jõudluse aspektist töötab rakendus rahuldavalt, Kõige kiiremini töötas rakendus libGDX raamistikuga. Teiseks valikuks oleks olnud libGDX raamistik, aga selle kasutamise ja dok.-is esinenud vananenud andmete tõttu kahtleb autor rakenduse aktuaalsuses.

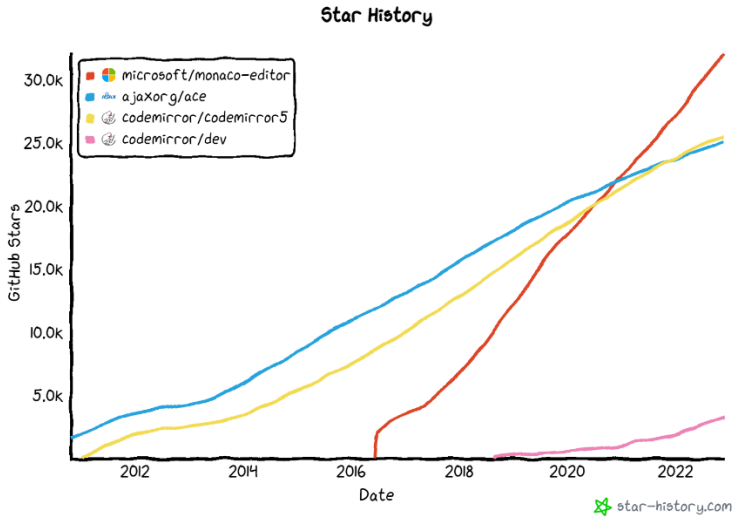
Antud töö raames loodav programm kirjutatakse kasutades PixiJS raamistikku. Programmi arenduseks kasutatakse JavaScript prog.k.-e otse kasutamise asemel TypeScript abstraktsioonikihti koodi vigade vältimiseks.

4.3 Tehnoloogia analüüs II

Nüüd on rakenduse keel ja kasutaja skriptide loomise keel välja selgitatud. Järgnevalt on vaja selgitada parimad JavaScript/TypeScript teegid erineva funktsionaalsuse tagamiseks.

4.3.1 JavaScript põhiste koodiredaktorite võrdlus

Järgnevalt võrreldakse koodiredaktori teeke kasutaja skriptide kirjutamise tagamiseks. Tegu ei ole rakenduse arendustööriistaga, vaid kasutajate koodiredaktoriga, mille eesmärk on skriptide loomise võimekuse tagamine. Selleks eksisteerib kolm põhilist teeki nimedega CodeMirror, Ace ja Monaco Editor. Kõik kolm koodiredaktorit on populaarsed, aga Monaco Editor on nendest uusim ja populaarseim (Joonis 5). CodeMirror teegi versioone 5 ja 6 käsitletakse võrdluse käigus eraldi.



Joonis 5. JavaScript põhiste koodiredaktorite populaarsus [42].

Võrdluses hinnatakse teegi üldist kasutajakogemust, seadistamise vajadust ja dokumentatsiooni (Tabel 6). Hea kasutajakogemus on ilmselgne vajadus. Seadistamise

vajaduse all hinnatakse töötava tekstiredaktori saamise kiirust, et vältida arenduskiiruse takerdumist erinevate konfigureerimise ülesannete taha. Dok. all hinnatakse juhendite, näidete ja demode kvaliteeti teegi kasutamiseks. Lisaks on välja toodud teegi arenduses kasutatud prog.k.-t, aga valiku tegemisel seda arvesse võetud ei ole. Võrdluse tulemused on kajastatud vastavalt algse analüüsi värviskeemile. Lisaks võetakse arvesse teekide populaarsuse kasvu, et tagada nende tulevikukindlus.

Tabel 6. JavaScript põhiste koodiredaktorite võrdlus.

Teegi nimi	**Keel**[1]	**Kasutajakogemus**	**Seadistamise vajadus**	**Dok.**
CodeMirror 5	JS	Otsingu funktsionaalsus jääb soovida, taane märgid puuduvad.	Täis kasutajakogemuse saamiseks peab kulutama aega erinevate moodulite lisamiseks.	Rahuldav juhend, demod ei kajasta hästi teegi võimekust.
CodeMirror 6	TS	Taane märgid puuduvad.	Autoril ei õnnestunud kolme tunniga teeki täielikult tööle saada. Nõuab palju seadistamist.	Liiga põhjalik juhend, näiteid ei ole.
Ace	JS	Kasutajaliides näeb natuke aegunud välja.	Soovitatava teegi saab tänu dok. interaktiivne teegi seadistuste demo üpriski kiiresti valmis.	Hea otsekohene juhend ja interaktiivne seadistuste demo.
Monaco Editor	TS	Võrratu! Töötab sarnaselt Visual Studio Code koodiredaktorile ja sisaldab koodi kontekstipõhist lõpetamist.	Vajab spetsiaalset konfigureerimist pakkimissüsteemiga. Soovitatav kasutada Webpack süsteemi.	Puudulik

Antud teekidest osutus kasutajakogemuse ja populaarsuse osas parimaks Monaco Editor. Teistest võrreldud koodiredaktoritest erineb Monaco Editor koodi kontekstipõhise lõpetamise (IntelliSense) võimekuse poolest. Kahjuks nõuab Monaco Editori kasutamine omakorda Webpack moodulipakkija kasutamist. Samuti ei saa seda dok. puudumise tõttu kergelt seadistada, aga vaikeväärtustega koodiredaktori kasutamine on võrratu kogemus.

[1] Teegi arenduses kasutatud programmeerimise keel (TS – TypeScript, JS - JavaScript).

4.3.2 JavaScript raja leidmise algoritmide võrdlus

Tabelite vaheliste suhtejoonte joonistamiseks kasutatakse raja leidmise algoritmi. Selleks on arendatud erinevaid JavaScript teeke nagu PathFinding.js, javascript-astar, rot.js, astar-typescript, EasyStar.js. Paraku ei toeta PathFinding.js, rot.js ja astar-typescript teegid graafi kaaludega tippe. Allesjäänud javascript-astar ja EasyStar.js teegid ei toeta teeleidmise lõpetamist etteantud funktsiooniga, et oleks võimalik kasutada A* algoritmi kindla sihiga ja mitme võimaliku lõpp-punktiga.

Nimetatud miinuste tõttu valmis raja leidmise teeki rakenduses kasutatud ei ole. Antud rakenduses kasutatakse täpsemalt peatükis 5.2.4 kirjeldatud spetsiifilist algoritmi.

5 Arendusprotsess ja valminud rakendus

Järgnevalt kirjeldatakse töö praktilist osa. Selles peatükkis selgitatakse rakenduse ülesehitust, arenduse käigus kasutatud koodimustreid, valminud rakendust ja selle edasiarenduse võimalusi.

5.1 Rakenduse ülesehitus

Rakenduse arenduseks valis autor TypeScript prog.k.-e ja PixiJS raamistiku. Tulenevalt andmete hoiustamise vajaduse puudumisest ja skriptimise turvalisuse tagamiseks on rakendus kirjutatud staatilise leheküljena. See tähendab, et rakendus ei kasuta serveripoolset komponenti sisu haldamiseks. Sellist lehekülge on lihtne erinevates veebimajutusteenustes üleval hoida. Veebimajutuseks kasutatakse tasuta Github Pages teenust, sest Github koodihoidlaga on autor varasemalt tuttav.

5.1.1 Rakenduse struktuur ja veebimajutus

Rakendus on kirjutatud TypeScript prog.k.-es. Rakenduse sõltuvuste haldamiseks kasutatakse Node Package Manager (NPM) mooduli haldajat. Rakenduse ehitamiseks peab TypeScript failid kompileerima JavaScript ES6/ES2015 mooduli failideks. Selle teisenduse tegemiseks kasutatakse Webpack moodulipakkijat, sest TypeScript kompilaator ise ei saa läbi NPM installitud väliste teekide importimisest aru. Webpack moodulipakkija konfiguratsioonifail ei toeta otse ES6 mooduleid, seega kasutatakse koos Webpack pakkijaga Babel ühilduvuskihti. Lisaks TypeScript failidele käsitleb Webpack veel Monaco Editori stiilifaile. Käsureal käsu "npm run build" käivitamise tulemusena kompileeritakse kõik TypeScript failid üheks JavaScript kokkupakitud failiks. Rakenduse käivitamiseks arenduskeskkonnas tuleb kasutada lokaalset serverit loovat tööriista, autor kasutas Python http.server moodulit.

Projekti haldamiseks kasutatakse Git versioonihaldustarkvara koos Github koodihoidlaga. Projekti hoitakse Github koodihoidla (Lisa 6) internetiaadressil. Kompileeritud JavaScript koodifailidega projekti risustamise vältimiseks ei hoita JavaScript faile otse Github koodihoidlas. Rakenduse Github keskkonda salvestamisel

käivitatakse Github serveris ".github/workflows/main.yml" failis defineeritud Github Action automatiseerimise töörist. Github Action loob ajutise virtuaalse konteineri, kuhu paigaldatakse Node süsteem, ehitatakse rakenduse JavaScript fail ja saadud tulemus salvestatakse Github projekti "gh-pages" harule. Github on seadistatud antud haru sisu esitama tasuta läbi Github Pages staatilise saidi veebimajutusteenuse (Lisa 6) internetiaadressil.

5.1.2 Rakenduse failistruktuur

Rakenduse struktuur on autori enda loodud, aga see peaks olema sarnane tüüpilisele TypeScript projektile. Antud rakenduse struktuur erineb eelkõige Github Action teenuse, Webpack pakkija ja rakenduse sisus kasutatud nimekonventsioonide poolest. Rakenduse arenduses kasutatud failide ja kaustade struktuuri kirjeldus:

- .github/workflows – kaust, mis sisaldab Github Action skriptifaili projekti ehitamiseks, kui kood Github koodihoidlasse üles laetakse
- node_modules – projekti lokaalsel arendusel kasutatud väliste teekide ajutine hoiustamiskaust
- webapp – peamine rakenduse kaust, mille sisu Github Pages veebimajutusteenus esitab:
    - pages – rakenduse terviklikud ja osalised HTML lehekülgede failid
    - wwwroot – lehekülgede töötamiseks vajalikud ressursi failid
    - ts – TypeScript failid rakenduse interaktiivsuse tagamiseks, mis kompileeritakse JavaScript failiks ja pannakse wwwroot kausta
- .gitignore – Git versioonihalduse poolt ignoreeritud failide ja kaustade nimekiri, mida Github koodihoidlasse üles ei laeta, näiteks node_modules kaust
- LICENSE – rakenduse MIT litsents
- package-lock.json – paigaldatud väliste teekide täpseid versioone lukustav automaatselt genereeritud fail
- package.json – inimloetav fail projekti väliste teekide haldamiseks ja projekti metaandmete ning standardiseeritud käsurea käskude hoiustamiseks
- pyServer.py – lokaalse serveri loomise Python skript
- README.md – versioonihaldusteenuses kuvatav projekti kirjeldusfail
- tsconfig.json – TypeScript konfiguratsioonifail, mida Webpack kasutab

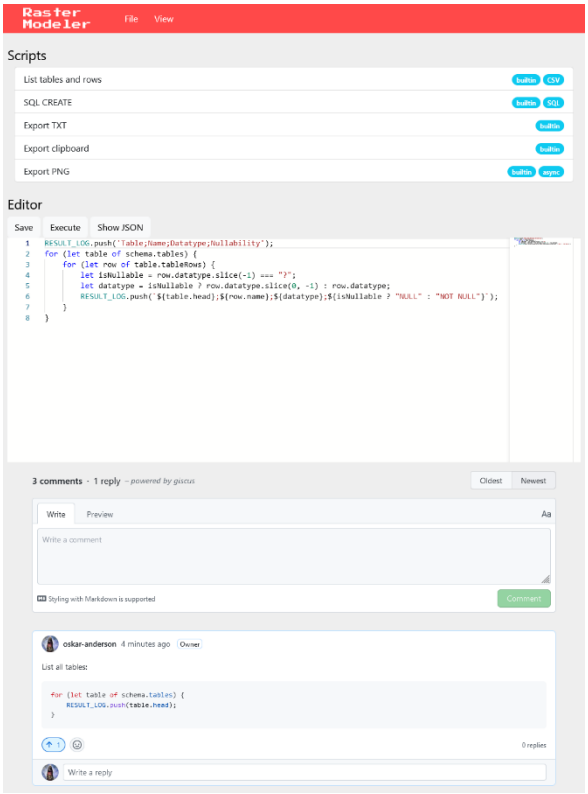
- webpack.config.babel.js – projekti ehitamiseks vajalik konfiguratsioonifail, mida kasutavad Webpack ja Babel koodi pakkijad

Lisaks koostab Github Action .nojekyll nimelise tühja faili Github Pages majutusteenuse juur kausta. Faili olemasolul ei ignoreeri Github Pages vaikeväärtusena alakriipsuga algavaid faile ja kaustu.

5.1.3 Rakenduse kasutajaliides

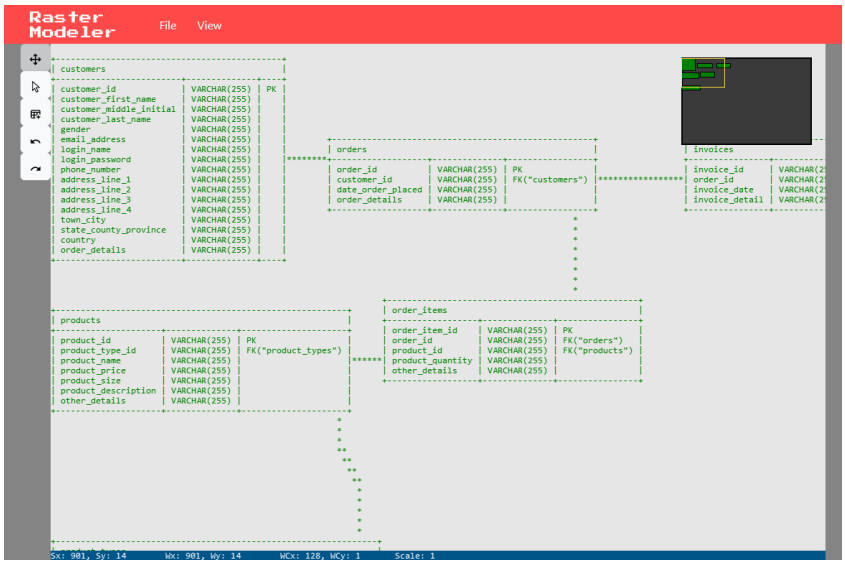
Rakenduse kasutajaliides koosneb PixiJS raamistiku sisseehitatud graafilistest elementidest ja harilikest HTML elementidest.

Üldiselt eelistati kasutajaliidese loomisel kasutada HTML elemente, sest see on lihtsam, funktsionaalsem ja üldiselt parem praktika [43]. Enamus rakenduse interaktiivsetest komponentidest nagu menüüribad ja *modal* aknad on loodud HTML elementidega. Skriptimise vaade on ehitatud üles puhtalt HTML elementidest (Joonis 6). Rakendus kasutab skriptide loomiseks Monaco Editor tekstiredaktori teeki ja nende jagamiseks giscus kommenteerimise teeki.



Joonis 6. Skriptide koostamise vaade.

Rakendus kasutab skeemi kuvamiseks *PIXI.BitmapText* klassil põhinevat tekstikuju formaati (Joonis 7). Navigeerimise miniakna elemendi ja ressursside laadimisriba kuvamiseks kasutatakse *PIXI.Graphics* klassi.



Joonis 7. Skeemide koostamise vaade.

PixiJS raamistik omab erinevaid kasutajaliideste loomiseks arendatud väliseid teeke nagu gown.js, EZGUI, PuxiJS, PIXI.TextInput ja pixi-viewport. Kahjuks on esimesed neli kasutajaliidese teeki aegunud ja ei ühildu viimase PixiJS versiooniga. Teek pixi-viewport ei toeta graafilisi kasutajaliidese elementide loomist, aga sellegipoolest osutub see rakenduses funktsionaalselt vajalikuks skeemi liigutamiseks ja suumimiseks.

5.2 Rakenduse arenduses kasutatud koodimustrid

Järgnevalt kirjeldatakse erinevaid rakenduse arenduses kasutatud koodimustreid ja algoritme. Koodimustreid on vaja rakenduse struktuuri tagamiseks, kindla funktsionaalsuse realiseerimiseks või prog.k.-est tulenevate puuduste tõttu.

5.2.1 Üldine rakenduse ülesehitus läbi olekumasina mustri

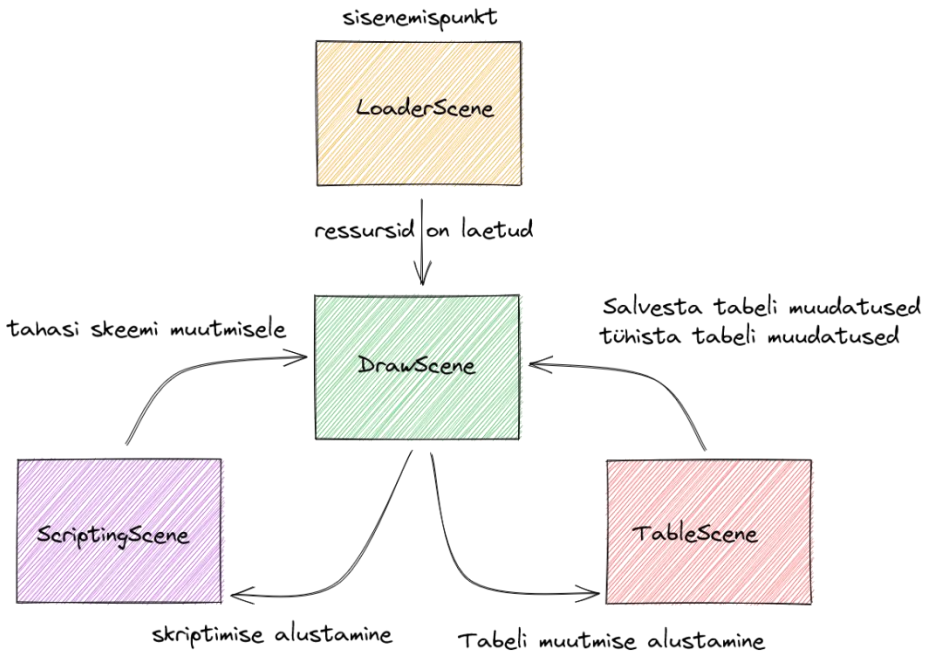
Rakenduse ülesehitus põhineb PixiJS raamistiku *PIXI.Container* klassil, mille ülesanne on grupeerida ekraanil kuvatavaid *PIXI.DisplayObject* elemente terviklikuks üksuseks. Selline ülesehitus lubab arendajal laiendada *PIXI.Container* klassi, et luua terviklikke ja iseseisvaid ekraani stseeni olekuid. Stseene on hallatud läbi lõpliku olekumasina mustri. Lisaks stseenidele on selle mustriga hallatud kasutaja hiire sündmusi vastavalt aktiivsele tööriistale.

Mustri põhimõte on grupeerida rakenduse oleku elemendid, millest ainult üks saab korraga aktiivne olla, ühe liidese ja välja alla. Mustri eesmärk on tagada head koodistruktuuri olekute ja nende muutuste haldamiseks. Mustri kasutamine aitab vältida suurte ja raskesti hallatavate tingimuslausete blokkide tekkimist. Olekud võivad olla üksteisest teadlikud ja alustada üleminekuid teistesse olekutesse läbi oleku haldaja klassi [44].

Mustri põhimõte on rakenduses realiseeritud *Manager* klassiga, mille *changeScene* meetod kontrollib rakenduse stseeni muutumist. Kõige esimeseks stseeniks on *LoaderScene*. Antud stseeni eesmärk on kuvada kasutajale rakenduse laadimisriba, alustada taustal rakenduse töötamiseks vajalike ressursside laadimist, sisaldada andmeid järgneva *DrawScene* stseeni loomiseks ning viia rakendus uuele stseenile läbi *Manager.changeScene* funktsiooni. Antud funktsionaalsuse koodiga saab tutvuda peatükis Lisa 2.

Olekumasina muster tagab hea koodi struktuuri läbi ülesannete põhise stseenide eralduse, sest iga stseen keskendub ainult enda rollidele. Lisaks saab vajadusel stseene jagada omakorda väiksemateks olekuteks nagu on tehtud *DrawScene* stseenis hiire sündmuste haldamiseks vastavalt aktiivsele tööriistale. Tööriistad täidavad erinevaid hiirega skeemi oleku muutmise rolle näiteks kaamera liigutamine, tabelite liigutamine, tabelite loomine ja tabelite valimine.

Antud rakenduse raames on stseeni olekutemuster autori arvates lihtsasti aru saadav, kergesti loetav ja kergesti visualiseeritav (Joonis 8).



Joonis 8. Rakenduse olekumusteri skeem.

5.2.2 *Undo/Redo* funktsionaalsuse realiseerimine läbi käsumustri

Rakenduse kasutamise lihtsustamiseks peab olema võimalik rakenduse oleku muudatusi tagasi võtta ehk lubada *undo/redo* operatsioone. Selline võimekus lubab kasutajal erinevaid operatsioone tühistada ja taastada rakenduse olek ennem operatsiooni tegemist.

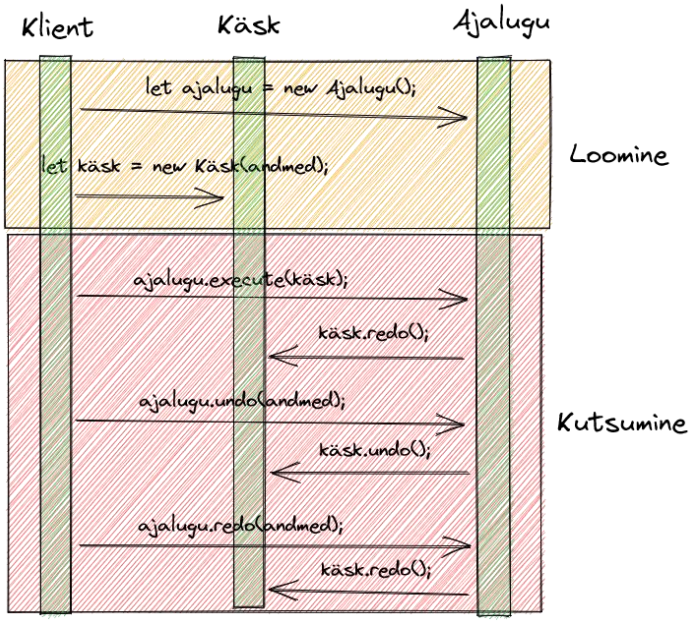
Raamat „Design Patterns: Elements of Reusable Object-Oriented Software“, mille autoriks on neli C++ tarkvarainseneri hüüdnimega „Gang of Four“ ehk GoF, toob välja 23 erinevat struktuuri-, käitumis- ja loomis disainimustrit programmeerimises korduvate probleemide lahendamiseks. Raamat väidab, et *undo/redo* funktsionaalsuse realiseerimiseks saab kasutada käsumustrit või teise nimega transaktsioonimustrit. Mustri struktuur sisaldab termineid: klient, käsk, vastuvõtja ja kutsuja. Mustri põhimõte on kliendis ehk üldises rakenduses luua vastuvõtja objekt, mis sisaldab reaalset rakenduse olekut ja funktsionaalsust ning käsu objekt, millele antakse vastuvõtja objekt konstruktori parameetrina kaasa. Käsk sisaldab *execute* käivitusmeetodi, mis kutsub konkreetset vastuvõtja funktsiooni. Käsu objekt antakse edasi kutsuja objektile, milleks on enamasti kasutajaliidese interaktiivne element nagu nupp, mis võib valmis käsu igal ajal käivitada. Käsumuster on põhimõtteliselt alternatiivne lähenemine objekt orienteeritud viisil tagasikutse funktsiooni loomisele. Raamat soovitab *undo/redo* funktsionaalsuse realiseerimiseks luua *unexecute* meetod ja hoida käivitatud käske ajaloo nimekirjas koos

nimekirja viimase aktiivse käsu indeksiga. Kahjuks ei tooda konkreetset näidet ja ei kirjeldata täpselt käsu andmete hoiustamist [45].

Antud rakenduses *undo/redo* funktsionaalsuse loomisel tuleb traditsioonilist käsumustrit mõningal määral muutma:

- Traditsiooniline muster on väga abstraktne. Käsk ainult delegeerib tööd - käsule antakse loomisel kaasa vastuvõtja objekt, mille konkreetset funktsiooni käsu käivitamisel kutsutakse. Liigne abstraktsioon ja üldsus teeb rakenduse mõistmise keerulisemaks ning arendamise aeganõudvamaks. Praktilisuse eesmärgil on vastuvõtja objekti andmed liigutatud käsu konstruktorisse, käsu käivitusmeetod on pandud realiseerima konkreetseid operatsioone ja vastuvõtja objekt on ära kustutatud.
- Nimetada kutsuja klass ümber ajaloo klassiks, mis hoiab täidetud käske JSON formaadis käsu nime ja argumentidega. See laseb käske hiljem tagasi võtta ja võimaldab vajadusel rakenduse käsu ajalugu eksportida. Miinusena peab käsu klasse eraldi registreerima, et neid hiljem nime järgi üles leida.
- Kasutada käsu ajaloo nimekirja ja indeksi asemel kasutatakse kahte pinupõhist nimekirja *undoHistory* ja *redoHistory*. See muudatus läheb paremini kokku rakenduse arendamiseks valitud JavaScript prog.k. sisseehitatud nimekirja funktsioonide puudulikkusega.
- Käsukude kutsumine käib läbi ajaloo objekti *execute* meetodi, mis enne käsu käivitamist lisab selle käsu ajaloo nimekirja. Käsu *execute* ja *unexecute* meetodid on nimetatud ümber *undo* ja *redo* meetoditeks. Ajalugu sisaldab oma *undo* ja *redo* meetodeid, mis kustuvad käskude *undo* ja *redo* meetodeid vastavalt ajaloo nimekirja olekule.

Joonis 9 selgitab rakenduses kasutatava käsumustri pseudokoodi.



Joonis 9. Rakenduses kasutatav käsumustri pseudokood.

Saadud koodimuster realiseerib *undo/redo* operatsioone hästi ja töötab selle rakenduse raames paremini kui GoF raamatus väljatoodud käsumuster. Rakenduses kasutatud mustri koodi koos konkreetse *CommandMoveTableRelative* saab näha peatükis Lisa 3.

5.2.3 Objektide konstrueerimine läbi koopia konstruktori

Eelmise mustri juures tuleb olla, tänu rakenduse arendamiseks osutunud TypeScript keele andmetüüpide teisenduse problemaatilisusest, tähelepanelik objektide andmetüüpide manipuleerimisel. JSON kujule mitte ühilduvate andmete (nagu meetodite) originaalkujule tagasi saamine on kergeks komistuskohaks. Selle probleemi vältimiseks võetakse kasutusele koopia konstruktor.

Andmetüübist tulenevate vigade vältimiseks saab kasutada *typestack/class-transformer* välist teeki või kirjutada käsitsi kõik objekti kopeeritavad andmeväljad uuele objektile ümber. Seoses välise teegi dok. aegumise, tööle saamise keerukuse ja liigse sõltuvuse vältimise eesmärgil osutas autor käsitsi ümberkirjutamise lahenduse kasuks [46].

Andmete ümberkopeerimise lihtsustamiseks on võetud kasutusele koopia konstruktori võte. Antud võtte põhimõte on koostada staatiline meetod, mis võtab sisendiks antud klassi objekti ja tagastab selle objekti sügava koopia (antud objekti väärtused ei viita olemasolevate objektide väärtustele) läbi klassi konstruktori [44]. Lisaks JSON andmete tagasi objektiks teisendamisele on meetodist kasu kasutajaliideses hõljuvate objektide

kujutamisel ilma originaalset objekti muutmata. Koopia konstruktori koodi näitega saab tutvuda peatükis Lisa 4.

5.2.4 Tabelite suhtejoonte kujutamine läbi A* algoritmi

Andmebaasi skeemid koosnevad tabelitest ja nende suhetest. Tabelite vahelisi suhete kujutamiseks on kasutatud jooni, mis saadakse läbi raja leidmise algoritmi. Joonte kuvamiseks on valitud A* raja leidmise algoritm, et tagada rakenduse jõudlus ja skeemi selgus. Kui näiteks MySQL Workbench rakenduses jooksevad tabelite vahelised jooned läbi tabelite, siis loodavas rakenduses on selliseid olukordi välditud.

Kasutatav algoritm peab leidma kiiresti läbitava teekonna kahe punkti vahel. Leitud teekond ei pea olema kõige otsem, vaid lihtsasti läbitav. Selleks kasutatakse ruudustikstruktuuri, kus iga ruut sisaldab enda läbimise hinda. Ruudustikku võib pidada graafi erijuhuks, kus graafi tipud asetsevad ühtlaselt ristkülikukujuliselt koordinaattasandikul. Ruudu hinna vaikeväärtus on 1, hind kasvab tabelite lähenduses, et tagada selgus tabeliga ühendatud joonte ja tabelist mööduvate joonte vahel. Tabelite asukohal paiknevate ruutude hind on 0 ehk läbimatu, mittepositiivse hinnaga läbitavaid ruute ei ole.

Raja leidmiseks on erinevaid algoritme. Üks lihtsamaid nendest on laiuti otsing. Antud algoritm leiab teekonna alguspunktist lõpp-punkti uurides algusest laialivalguvalt kõigis suunas võrdselt. Sellise laienemise saavutamiseks kasutatakse esimesena sisse, esimesena välja jada. Loodavasse rakendusse antud algoritm ei sobi, sest see laieneb ühtlaselt võtmata arvesse ruutude hindasid ja on aeglane [47].

Hindade arvestamiseks saab kasutada muutumatu hinna otsingu algoritmi (*Uniform-Cost Search*). Antud algoritm on variatsioon Dijkstra algoritmist, mis on tööprotsessis konkreetsem ja jõudluses optimaalsem [48]. Antud algoritm kasutab teekonna otsimiseks prioriteedi jada (*priority queue*), mis organiseerib otsimisprotsessi vastavalt alguspunktist ruudule jõudmise hinnale. Antud algoritm on aeganõudev, sest see ei laiene sihiga lõpp-punkti [47].

Muutumatu hinna otsingu algoritmi optimeerimiseks saab prioriteedi jada laiendada lisades alguspunktist antud ruudule jõudmise hinnale juurde heuristilise funktsiooni hinna. Saadud funktsioon kannab nime A* algoritm (häädatakse A-täht või ing. *A-star*).

Heuristiline funktsioon tagastab kahe punkti vahelise vahemaa eeldatud hinna. Rakenduses kasutatakse vahemaa arvutamiseks Manhattani kauguse (ing. *Manhattan distance*) ja linnulennu kauguse (eukleidiline kaugus, ing. *Euclidean distance*) funktsioone (Joonis 10) [47], [49].

```
static euclidean(a:{x:number, y:number}, b:{x:number, y:number}) {
      let dx = Math.abs(a.x - b.x);
      let dy = Math.abs(a.y - b.y);
      return Math.sqrt(dx * dx + dy * dy);
};

static manhattan(a:{x:number, y:number}, b:{x:number, y:number}) {
      return Math.abs(a.x - b.x) + Math.abs(a.y - b.y);
};
```

Joonis 10. Eukleidilise kauguse ja Manhattani kauguse TypeScript kood.

Linnulennu kaugust kasutatakse tabelite vaheliste suhtejoonte joonistamise järjekorra ja nende joonte alguspunktide määramiseks. Suhtejoonte lõpp-punkte on palju, aga A* algoritmi sihitud lõpp-punkt asetseb lõpptabeli ristkülliku suurima ühise algustabelile lähima ruudu keskel. A* funktsioon ei jõua kunagi sihitud tabeli lõpp-punkti, vaid tagastab sihitud tabeliga kokkupuutel leitud teekonna. A* heuristikana kasutatakse Manhattani kaugust, sest see kajastab joonte võimalikku põhi, ida, lõuna, lääs neljasuunalist liikumist.

Algoritmi on tehtud väikene muudatus, et saavutada ilusam joonte paiknemine. Vastasel juhul saadakse liigsete kõverustega või ebainimliku paiknemisega jooned, mis häirivad skeemi üldist loetavust.

> *The most common question I get when people run pathfinding on a grid is why don't my paths look straight?*
>
> *— Amit Patel, Red Blob Games [50]*

Algselt proovis autor arvestada prioriteedi jadas elementide lisamise järjekorda. Vastasel juhul väljuvad sama prioriteediga elemendid jadast määratlemata järjekorras ja see põhjustab ebavajalikke joonte pöördeid. Antud pöörded ei vähenda raja pikkust ruudustikstruktuuris, aga vähendavad skeemi visuaalset selgust. Sirgete joonte saavutamiseks peab ruudustikstruktuuri graafi tippude naabrid olema võimalik tagastada konkreetses järjekorras ja seda järjekorda tuleb säilitada prioriteedijadas, selleks tuleb igale sisestatavale elemendile lisada väikene järkjärguline boonusprioriteet. Antud

lähenemine töötab sirgete joonte tagamiseks, aga need jooned näevad ebainimlikud välja, sest nad jooksevad otse takistuste poole ja hakkavad seejärel takistuse ümbert mööda minema. Antud olukorda saab vältida Amit Patel poolt pakutud siksakilise diagonaalse liikumisega.

Amit Patel soovitab jooksvalt muuta ruudustikstruktuuri elementide hinda naaberelementidest laienemisel, et tagada siksakiliselt diagonaalsed jooned. See tagab prioriteedi jada elementide väljumise konkreetses järjekorras olenevalt ruudule jõudmise suunast kabelaua (x + y) % 2 mustris. Selle tulemusena muutub A* joonte paiknemine ootuspärasemaks, sest jooned tunduvad takistusi vältima rohkem ja loomulikumalt [50].

Rakenduses kasutatud A* algoritm on välja toodud Lisas 5.

5.3 Valminud rakendus

Antud töö raames valmis unikaalne veebirakendus, mille sarnast olemasolevate rakenduse võrdluses tuvastada ei õnnestunud. Rakendus vastab kõigile loodava rakendusele seatud nõuetele. Lisaks defineeritud nõuetele loodi rakenduse ekraani asukoha tuvastamiseks ja skeemil navigeerimiseks miniakna komponent ning parema kasutajakogemuse tagamiseks sisaldab rakendus käskude ajalugu, mis lubab kasutaja poolt tehtud muudatusi tagasi võtta.

Rakenduse arenduse suurimaks probleemiks osutus rakenduse jõudlus. Suurt jõudlust on vaja brauseri poolt kuvatavate elementide sujuvaks muutmiseks kasutaja poolt. Jõudlus esineb peamiselt ekraani akna liigutamise ja suumimise kiiruses ning võimes kuvada tabelite liigutamisel ja loomisel nende elementide sujuvat hõljumist kursori positsioonil. Tänu PixiJS raamistiku kasutamisele suudab rakendus, testimise käigus nõrgemaks osutunud Mozilla Firefox brauseris, korraga sujuvalt kuvada umbes 70000 graafika objekti. Kõrgematel ekraani suumi tasemetel on ekraani aknast väljapoole jäävad elemendid peidetud.

Projekti eesmärk kasutada skeemi andmete esitamiseks tekstipõhist formaati õnnestus. See lihtsustab skeemi andmete muutmist ja lubab skeeme teksti kujul Markdown ja HTML koodiblokkide sees kujutada.

Rakendus sisaldab JavaScript skriptimiskeelt kasutaja loodud skriptide käivitamiseks. Skriptide kirjutamiseks on rakenduses kasutusel Monaco Editor tekstiredaktor. Skriptimise töötamist on tõestatud SQL loomispäringuid genereeriva sisseehitatud skriptiga. Rakendus on avalikustatud avatud lähtekoodiga, et innustada kasutajatel skriptide loomist. Kasutajad saavad skripte omavahel jagada läbi giscus kommenteerimise teegi.

Antud rakendus on tasuta kasutatav Lisa 6 aadressil.

5.4 Edasiarenduse võimalused

Järgnevalt kirjeldatakse rakenduse funktsionaalsuse edasiarenduse võimalusi.

Rakendus võiks võimaldada skeemi koostamist olemasoleva andmebaasi põhjal. Paljud konkureerivad rakendused toetavad seda funktsionaalsust ja seega tasuks nendest eeskuju võtta. Skeemi importimiseks tuleb kõigepealt tuvastada parim formaat andmebaasi mudeli eksportimiseks. Arvatavasti oleks selleks SQL loomispäringute formaat, aga ennem tasuks uurida Lucidchart skeemi importimise lähenemist.

Rakenduse skriptimise võimalust saaks edasi arendada sisseehitatud skriptide koguse suurendamisega. See aitaks tuvastada miinuseid skriptide loomise protsessis ja aitaks luua skriptide korduvate operatsioonide vältimiseks globaalseid funktsioone. Skriptimist saaks laiendada näiteks dok.-i, ORM mudeleid ja andmebaasi tabelite kasutajaliidese vaateid genereeriva skriptiga.

Dokumentatsiooni genereeriva skripti juures peaks tabelitele ja nende väljadele olema võimalik selgitavaid kirjeldusi jätta. Hetkel ei ole selgituste skeemile salvestamine võimalik. Selle võimaldamiseks tuleb skeemi salvestusele lisada juurde ekstra JSON andmed. Antud andmed ei oleks skeemil näha ja neid kasutataks ainult skriptides.

Skriptide arvu suurenedes võib osutuda vajalikuks skriptide organiseerimine. Skripte peaks olema võimalik otsida nimeliselt ja vastavalt kategooriale. Hetkel puudub võimalus kasutajal skripti kategooria määramiseks.

6 Kokkuvõte

Käesoleva bakalaureusetöö põhieesmärgiks oli asendada autori poolt varem andmebaasi mudelite visualiseerimiseks kasutatud QSEE SuperLite rakendus parema alternatiivse rakenduse vastu välja. Rakenduse eesmärk on lisaks andmebaasi mudelite kujutamisele lasta kasutajal programmi väljundit enda loodud skriptidega kergesti laiendada ja esitada ning hoida andmeid tekstikujulises formaadis.

Töö käigus tuvastati QSEE SuperLite programmi puuduseid ja võrreldi teisi andmebaasi skeemide visualiseerimiseks mõeldud rakendusi. Sobiva alternatiivse rakenduse puudumise tõttu otsustas autor arendada enda rakenduse. Töö analüüsis määratakse loodava rakenduse nõuded ja selgitatakse rakenduse arenduses kasutatud graafika kuvamist lubava raamistiku ja programmeerimiskeele valikut vastavalt seatud nõetele. Edasi kirjeldatakse töö praktilist osa, mis hõlmab rakenduse ülesehitust, kasutajaliidese arengus tehtud otsuseid, kasutatud koodimustrite kirjeldust, valminud rakendust ja selle edasiarendamise võimalusi.

Projekt õnnestus, sest valminud rakendus vastab analüüsi käigus seatud nõuetele. Rakendus on skriptidega kergesti laiendatav, töötab kiiresti, kasutab rakenduse oleku kuvamiseks tekstipõhist formaati, on avatud lähtekoodiga (Lisa 6) ja on internetist avalikult kättesaadav (Lisa 6). Valminud rakendusega on võimalik koostada andmebaasi skeeme, neid eksportida teksti ja pildi kujul, kirjutada skripti faile ning käivitada neid skeemi andmete põhjal. Bakalaureusetöö tulemusel on nüüd võimalik tasuta andmebaasiskeeme veebis mugavalt visualiseerida ja rakendada skeemidele kasutaja loodud skripte.

Kasutatud kirjandus

[1] QSEE Technologies, „Modeling Tools From QSEE Technologies“. [Võrgumaterjal]. Saadaval: https://web.archive.org/web/20070427134104/http://www.qsee-technologies.com/index.htm. [Kasutatud 25.11.2022].

[2] diagrams.net, „diagrams.net Integrations”. [Võrgumaterjal]. Saadaval: https://www.diagrams.net/integrations. [Kasutatud 25.11.2022].

[3] Lucidchart, „Lucidcart plans” [Võrgumaterjal]. Saadaval: https://lucid.app/pricing/lucidchart. [Kasutatud 25.11.2022].

[4] MySQL, „MySQL Workbench Features”. [Võrgumaterjal]. Saadaval: https://www.mysql.com/products/workbench/features.html. [Kasutatud 25.11.2022].

[5] fabFORCE.net, „Overview”. [Võrgumaterjal]. Saadaval: https://www.fabforce.net/dbdesigner4. [Kasutatud 25.11.2022].

[6] Stack Overflow, „Questions tagged [mysql-workbench]”. [Võrgumaterjal]. Saadaval: https://stackoverflow.com/questions/tagged/mysql-workbench?tab=Votes. [Kasutatud 25.11.2022].

[7] DbSchema, „DbSchema Editions". [Võrgumaterjal]. Saadaval: https://dbschema.com/editions.html. [Kasutatud 25.11.2022].

[8] DbSchema, „Purchase DbSchema Pro”. [Võrgumaterjal]. Saadaval: https://dbschema.com/purchase.html. [Kasutatud 25.11.2022].

[9] Vertabelo, „Pricing”. [Võrgumaterjal]. Saadaval: https://vertabelo.com/pricing. [Kasutatud 25.11.2022].

[10] QuickDBD, „Pricing”. [Võrgumaterjal]. Saadaval: https://www.quickdatabasediagrams.com. [Kasutatud 25.11.2022].

[11] Monodraw, „Introduction”. [Võrgumaterjal]. Saadaval: https://monodraw.helftone.com/#introduction. [Kasutatud 25.11.2022].

[12] Javatpoint, „MySQL Describe Table”. [Võrgumaterjal]. Saadaval: https://www.javatpoint.com/mysql-describe-table. [Kasutatud 25.11.2022].

[13] K. O. Anderson, „Electron and PixiJS Text Performance”. [Võrgumaterjal]. Saadaval: https://github.com/oskar-anderson/ElectronAndPixiJSTextPerformance. [Kasutatud 25.11.2022].

[14] Star History, „Github Star History”. [Võrgumaterjal]. Saadaval: https://star-history.com/#pixijs/pixijs&dotnet/aspnetcore&libgdx/libgdx&dotnet/blazor&HaxeFlixel/flixel&Date. [Kasutatud 25.11.2022].

[15] PixiJS, „PixiJS”. [Võrgumaterjal]. Saadaval: https://github.com/pixijs/pixijs. [Kasutatud 25.11.2022].

[16] .NET Platform, „Blazor”. [Võrgumaterjal]. Saadaval: https://github.com/dotnet/blazor. [Kasutatud 25.11.2022].

[17] .NET Platform, „ASP.NET Core”. [Võrgumaterjal]. Saadaval: https://github.com/dotnet/aspnetcore. [Kasutatud 25.11.2022].

[18] HaxeFlixel, „HaxeFlixel”. [Võrgumaterjal]. Saadaval: https://github.com/HaxeFlixel/flixel. [Kasutatud 25.11.2022].

[19] K. O. Anderson, „Result depends on code outside script”. [Võrgumaterjal]. Saadaval: https://github.com/HaxeFoundation/hscript/issues/117. [Kasutatud 25.11.2022].

[20] Haxe Foundation, „hscript”. [Võrgumaterjal]. Saadaval: https://github.com/HaxeFoundation/hscript. [Kasutatud 25.11.2022].

[21] libGDX, „libGDX”. [Võrgumaterjal]. Saadaval: https://github.com/libgdx/libgdx. [Kasutatud 25.11.2022].

[22] slay_lines, „Canvas Engines Comparison”. [Võrgumaterjal]. Saadaval: https://benchmarks.slaylines.io/. [Kasutatud 25.11.2022].

[23] PixiJS, „What PixiJS Is”. [Võrgumaterjal]. Saadaval: https://pixijs.io/guides/basics/what-pixijs-is.html. [Kasutatud 25.11.2022].

[24] PixiJS, „Examples”. [Võrgumaterjal]. Saadaval: https://pixijs.io/examples. [Kasutatud 25.11.2022].

[25] PixiJS, „PixiJS — The HTML5 Creation Engine”. [Võrgumaterjal]. Saadaval: https://pixijs.download/release/docs/index.html. [Kasutatud 25.11.2022].

[26] AngelCode, „Bitmap Font Generator”. [Võrgumaterjal]. Saadaval: https://www.angelcode.com/products/bmfont. [Kasutatud 25.11.2022].

[27] A. Heddings, „What Is Microsoft’s Blazor Web Framework, and Should You Use It?” 2021. [Võrgumaterjal]. Saadaval: https://www.howtogeek.com/devops/what-is-microsofts-blazor-web-framework-and-should-you-use-it. [Kasutatud 25.11.2022].

[28] Microsoft, „ASP.NET Core Blazor”. 2022. [Võrgumaterjal]. Saadaval: https://docs.microsoft.com/en-us/aspnet/core/blazor. [Kasutatud 25.11.2022].

[29] K. O. Anderson, „Dotnet Performance Test”. [Võrgumaterjal]. Saadaval: https://github.com/oskar-anderson/DotnetPerformanceTest. [Kasutatud 25.11.2022].

[30] HaxeFlixel, „About”. [Võrgumaterjal]. Saadaval: https://haxeflixel.com/documentation/about. [Kasutatud 25.11.2022].

[31] Haxe Foundation, „History”. [Võrgumaterjal]. Saadaval: https://haxe.org/manual/introduction-haxe-history.html. [Kasutatud 25.11.2022].

[32] Haxe Foundation, „Compiler Targets”. [Võrgumaterjal]. Saadaval: https://haxe.org/documentation/introduction/compiler-targets.html. [Kasutatud 25.11.2022].

[33] K. O. Anderson, „Lime website URL has changed”. [Võrgumaterjal]. Saadaval: https://github.com/HaxeFlixel/flixel-docs/pull/255. [Kasutatud 25.11.2022].

[34] HaxeFlixel, „Documentation”. [Võrgumaterjal]. Saadaval: https://haxeflixel.com/documentation. [Kasutatud 25.11.2022].

[35] HaxeFlixel, „API”. [Võrgumaterjal]. Saadaval: https://api.haxeflixel.com. [Kasutatud 25.11.2022].

[36] K. O. Anderson, „HaxeFlixel Text Performance”. [Võrgumaterjal]. Saadaval: https://github.com/oskar-anderson/HaxeFlixelTextPerformance. [Kasutatud 25.11.2022].

[37] libGDX, „History”. [Võrgumaterjal]. Saadaval: https://libgdx.com/history. [Kasutatud 25.11.2022].

[38] libGDX, „Creating a Project”. [Võrgumaterjal]. Saadaval: https://libgdx.com/wiki/start/project-generation. [Kasutatud 25.11.2022].

[39] libGDX, „Demos & Tutorials”. [Võrgumaterjal]. Saadaval: https://libgdx.com/wiki/start/demos-and-tutorials. [Kasutatud 25.11.2022].

[40] libGDX, „Javadocs”. [Võrgumaterjal]. Saadaval: https://javadoc.io/doc/com.badlogicgames.gdx/gdx/latest/index.html. [Kasutatud 25.11.2022].

[41] K. O. Anderson, „libGDX BitmapText Performance”. [Võrgumaterjal]. Saadaval: https://github.com/oskar-anderson/libGDXTextPerformance. [Kasutatud 25.11.2022].

[42] Star History, „Github Star History”. [Võrgumaterjal]. Saadaval: https://star-history.com/#microsoft/monaco-editor&ajaxorg/ace&codemirror/codemirror5&codemirror/dev&Date. [Kasutatud 29.11.2022].

[43] Web Hypertext Application Technology Working Group, „HTML Living Standard”. [Võrgumaterjal]. Saadaval: https://html.spec.whatwg.org/multipage/canvas.html#best-practices. [Kasutatud 25.11.2022].

[44] A. Shvets, *Dive Into Design Patterns.* 2019.

[45] E. Gamma, J. Vlissides, R. Johnson, R. Helm, *Design Patterns: Elements of Reusable Object-Oriented Software*. Westford: Addison-Wesley, 1996. [E-book]. Loetud aadressil: http://www.javier8a.com/itc/bd1/articulo.pdf. [Kasutatud 25.11.2022].

[46] TypeStack, „class-transformer”. [Võrgumaterjal]. Saadaval: https://github.com/typestack/class-transformer. [Kasutatud 25.11.2022].

[47] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach,* 4th ed. Boston: Pearson, 2020.

[48] A. Felner, „Position Paper: Dijkstra’s Algorithm versus Uniform Cost Search or a Case Against Dijkstra’s Algorithm” in *Vol. 2 No. 1 (2011): Fourth Annual Symposium on Combinatorial Search,* Castell de Cardona, 2011. [Võrgumaterjal]. doi: https://doi.org/10.1609/socs.v2i1.18191. [Kasutatud 22.12.2022].

[49] I. Millington, J. Funge. *Artificial Intelligence for Games,* 2nd ed. Burlington: Morgan Kaufmann Publishers, 2009.

[50] A. Pattel, „Implementation of A*”. [Võrgumaterjal]. Saadaval: https://www.redblobgames.com/pathfinding/a-star/implementation.html. [Kasutatud 25.12.2022].

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

Mina, Karl Oskar Anderson

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Vabavaralise rakenduse arendamine andmebaasimudelite visualiseerimiseks", mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

29.12.2022

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 – Rakenduses PixiJS raamistiku käivitamine koos lühendatud olekumasina mustriga

```
// file Program.ts
import { Manager } from "./Manager";
import { LoaderScene } from "./Scenes/LoaderScene";
import { Draw } from "./model/Draw";

export class Program {

    static main(draw: Draw): void {
        Manager.initialize(1080, 720, 0xe6e6e6);
        Manager.changeScene(new LoaderScene(draw));
    }
}

// file Manager.ts
import { extensions, InteractionManager, Application, DisplayObject }
from "pixi.js";
import { EventSystem } from '@pixi/events';

export class Manager {
    static getRenderer () {
        return this.app.renderer;
    }

    private constructor() { }

    private static app: Application;
    private static currentScene: IScene;

    private static _width: number;
    private static _height: number;


    public static get width(): number {
        return Manager._width;
    }
    public static get height(): number {
        return Manager._height;
    }


    public static initialize(width: number, height: number,
background: number): void {
```

```
        Manager._width = width;
        Manager._height = height;

        extensions.remove(InteractionManager);
        Manager.app = new Application({
            backgroundColor: background,
            width: width,
            height: height
        });
        const { renderer } = Manager.app;
        renderer.addSystem(EventSystem, 'events');

        document.querySelector('.canvas-
container')!.appendChild(this.app.view);

document.querySelector('canvas')!.addEventListener('contextmenu', (e)
=> { e.preventDefault(); });
        document.querySelector('canvas')!.addEventListener('wheel',
function(event) { event.preventDefault(); });
        this.app.view.addEventListener("click", (event) => {
Manager.currentScene.mouseEventHandler(event) });
        this.app.view.addEventListener("mousedown", (event) => {
Manager.currentScene.mouseEventHandler(event) });
        this.app.view.addEventListener("mouseup", (event) => {
Manager.currentScene.mouseEventHandler(event) });

        Manager.app.ticker.add((time) => Manager.update());
    }

    public static changeScene(newScene: IScene): void {
        if (Manager.currentScene) {
            Manager.app.stage.removeChild(Manager.currentScene);
            Manager.currentScene.destroy();
            Manager.currentScene.destroyHtmlUi();
        }

        Manager.currentScene = newScene;
        Manager.app.stage.addChild(Manager.currentScene);
        newScene.init();
    }

    private static update(): void {
        if (Manager.currentScene) {
            Manager.currentScene.update(Manager.app.ticker.deltaMS);
        }
    }
}
```

```
export interface IScene extends DisplayObject {
    update(deltaMS: number): void;
    init(): void
    destroyHtmlUi(): void
    mouseEventHandler(event: MouseEvent): void
}
```

Lisa 3 – *Undo/Redo* funktsionaalsuse realiseerimise koos konkreetse *CommandMoveTableRelative* käsuga

```
// using the pattern from application
// call this on initialization, not for every command
this.draw.history = new History();
// ... some other code ...
// now is the time to execute the command
this.draw.history.execute(new CommandMoveTableRelative(
    this.draw, {
        id: hoverSource.id,
        x: xDiff,
        y: yDiff
    })
);
// undo/redo command like this
this.draw.history.undo(this.draw);
this.draw.history.redo(this.draw);


// file History.ts
import { Draw } from "../model/Draw";
import { CommandModifyTable } from "./appCommands/CommandModifyTable";
import { CommandMoveTableRelative } from
"./appCommands/CommandMoveTableRelative";
import { ICommand } from "./ICommand";


export class History {
    static implementations = [
        { name: CommandMoveTableRelative.name, constructor:
CommandMoveTableRelative },
        { name: CommandModifyTable.name, constructor:
CommandModifyTable },
    ];
    undoHistory: string[] = [];
    redoHistory: string[] = [];

    constructor() {}

    execute(commandInstance: ICommand<any>) {
        let command = { commandName: commandInstance.constructor.name,
args: commandInstance.args};
        if (! History.implementations.some(x => x.name ===
command.commandName)) throw Error(`Implementation not registered!
commandName: ${command.commandName}` );
```

```
        this.undoHistory.push(JSON.stringify(command));
        this.redoHistory = [];
        commandInstance.redo();
    }

    private getICommandInstance(command: CommandPattern, context:
Draw): ICommand<any> {
        let implementation = History.implementations.find(x => x.name
=== command.commandName);
        if (! implementation) throw Error(`Unknown command given!
commandName: ${command.commandName}`);
        return new implementation.constructor(context, command.args);
    }

    redo(context: Draw) {
        if (this.redoHistory.length === 0) return;
        let command = JSON.parse(this.redoHistory.pop()!) as
CommandPattern;
        this.undoHistory.push(JSON.stringify(command));
        this.getICommandInstance(command, context).redo();
    }

    undo(context: Draw) {
        if (this.undoHistory.length === 0) return;
        let command = JSON.parse(this.undoHistory.pop()!) as
CommandPattern;
        this.redoHistory.push(JSON.stringify(command));
        this.getICommandInstance(command, context).undo();
    }
}

interface CommandPattern {
    commandName: string;
    args: any;
}

// file ICommand.ts
import { Draw } from "../model/Draw";

export interface ICommand<T> {
    context: Draw;
    args: T;
    redo(): void;
    undo(): void;
}

// file CommandMoveTableRelative.ts
```

```
import { Draw } from "../../model/Draw";
import { ICommand } from "../ICommand";

export class CommandMoveTableRelative implements
ICommand<CommandMoveTableRelativeArgs> {
    context: Draw;
    args: CommandMoveTableRelativeArgs;

    constructor(context: Draw, args: CommandMoveTableRelativeArgs) {
        this.context = context;
        this.args = args;
    }

    redo() {
        let table = this.context.schema.tables.find(x => x.id ===
this.args.id)!;
        table.position.x += this.args.x;
        table.position.y += this.args.y;
    }

    undo()  {
        let table = this.context.schema.tables.find(x => x.id ===
this.args.id)!;
        table.position.x -= this.args.x;
        table.position.y -= this.args.y;
    }
}

export interface CommandMoveTableRelativeArgs {
    id: string; x: number; y: number;
}
```

Lisa 4 – Koopia konstruktori näide objektide konstrueerimiseks

```
export class Person {
    public firstName: string;
    public lastName: string;

    // Constructor overloading is problematic in TS - init
constructors are used instead
    private constructor({ firstName: string, lastName: string }) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    // Regular constructor
    static init(firstName: string, lastName: string) {
        return new Person({
            firstName: firstName,
            lastName: lastName
        });
    }

    // Copy constructor - can take JSON Person and not real Person
    static initClone(person: Person) {
        return new Person({
            firstName: person.firstName,
            lastName: person.lastName
        });
    }

    getFullName() {
        return this.firstName + " " + this.lastName;
    }
}
let john = Person.init("John", "Doe");
let parsedJohn = JSON.parse(JSON.stringify(john));
// console.log(parsedJohn.getFullName());  // this will not work
console.log(Person.initClone(parsedJohn).getFullName());  // John Doe
```

Lisa 5 – Rakenduses A* algoritmi realiseerimine

```
// file AStarFinderCustom.ts
import { PriorityQueue } from '@datastructures-js/priority-queue';
import { WorldGrid } from './WorldGrid';

export default class AStarFinderCustom {
    heuristic: (a: { x: number, y: number }, b: { x: number, y: number
}) => number;

    constructor(heuristic: ((a: { x: number, y: number }, b: { x:
number, y: number }) => number)) {
        this.heuristic = heuristic;
    }

    static euclidean(a:{x:number,y: number}, b:{x:number,y:number}) {
        let dx = Math.abs(a.x - b.x);
        let dy = Math.abs(a.y - b.y);
        return Math.sqrt(dx ** 2 + dy ** 2);
    };

    static manhattan(a: { x: number, y: number }, b: { x: number, y:
number }) {
        return Math.abs(a.x - b.x) + Math.abs(a.y - b.y);
    };

    /**
     * Find and return the path.
     * @return {Array<Array<number>>} The path, including both start
and end positions.
     */
    findPath(start: { x: number, y: number }, heuristicTarget: { x:
number, y: number }, ends: { x: number, y: number }[], grid:
WorldGrid) {
        if (ends.length === 0) return [];
        let frontier = new PriorityQueue<{ value: {x: number, y:
number}, cost: number}>
            ((
                a: {value: any, cost: number},
                b: {value: any, cost: number}
            ) => { return a.cost < b.cost ? -1 : 1 })   // lowest cost
will pop first
            .push({ value: start, cost: 0 });
        let cameFrom: Map<string, { x: number, y: number } | null> =
new Map()
            .set(grid.getPointId(start), null);
        let costSoFar: Map<string, number> = new Map()
```

```
            .set(grid.getPointId(start), 0);
        let end = null;
        while (! frontier.isEmpty()) {
            let current = frontier.pop().value;

            if (ends.findIndex(end => end.x === current.x && end.y ===
current.y) !== -1) {
                end = current;
                break;
            }

            for (let next of grid.neighbors(current.x, current.y)) {
                let startToNextCost =
costSoFar.get(grid.getPointId(current))! +
grid.getNeighborCost(current, next);
                if (! costSoFar.has(grid.getPointId(next)) ||
startToNextCost < costSoFar.get(grid.getPointId(next))!) {
                    costSoFar.set(grid.getPointId(next),
startToNextCost);
                    let priority = startToNextCost +
this.heuristic(next, heuristicTarget);
                    frontier.push({ value: next, cost: priority });
                    cameFrom.set(grid.getPointId(next), { x:
current.x, y: current.y });
                }
            }
        }

        if (end === null) { return []; }
        let route: { x: number, y: number }[] = [end];
        while (true) {
            let next = cameFrom.get(grid.getPointId(route[0]));
            if (!next) { break; }
            route.unshift(next);
        }
        return route;
    }
}

// file WorldGrid.ts
export class WorldGrid {

    width: number;
    height: number;
    nodes: { [id: string]: number };  // keys are from function
getPointId

    constructor(grid: number[][]) {
```

```
        this.height = grid.length;
        this.width = grid[0].length ?? 0;
        this.nodes = {};
        for (let y = 0; y < this.height; y++) {
            for (let x = 0; x < this.width; x++) {
                this.nodes[this.getPointId({ x: x, y: y})] =
grid[y][x];
            }
        }
    }

    inBounds(point: { x: number, y: number }) {
        return 0 <= point.x && point.x < this.width && 0 <= point.y &&
point.y < this.height;
    }

    passable(point: { x: number, y: number }) {
        return this.nodes[this.getPointId(point)] !== 0;
    }

    getPointId(point: { x: number, y: number}) {
        return `${point.y},${point.x}`;
    }

    neighbors(x: number, y: number): { x: number, y: number }[] {
        return [
            {x: x+1, y: y},
            {x: x-1, y: y},
            {x: x, y: y-1},
            {x: x, y: y+1}  // E W N S
        ]
            .filter((item) => this.inBounds(item))
            .filter((item) => this.passable(item));
    }

    getNeighborCost(orig: { x: number, y: number}, neighbor: { x:
number, y: number}): number {
        let cost = this.nodes[this.getPointId(neighbor)];
        let nudge = 0; // Manhattan diagonal path nudge by Amit Patel
        if ((orig.x + orig.y) % 2 == 0 && neighbor.x != orig.x) {
nudge = 1 }
        if ((orig.x + orig.y) % 2 == 1 && neighbor.y != orig.y) {
nudge = 1 }
        return cost + 0.0001 * nudge;
    }
}
```

Lisa 6 – Rakenduse enda ja selle koodihoidla viide

Rakendus – https://oskar-anderson.github.io/RasterModeler/pages/draw.html

Rakenduse koodihoidla – https://github.com/oskar-anderson/RasterModeler