TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Taaniel Sülla 185002IADB

Laste arengut toetavate mängude keskkonna “Nutitund” arendus

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2021

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: Taaniel Sülla

23.04.2021

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua toimiv avalik veebikeskkond milles on võimalik eelkooli lastel, kui ka esmaklassi lastel konstruktiivselt veeta aega nii mängides kui ka samaaegselt ennast arendades. Lahendus adresseerib konkreetselt lasteaiaõpetajate poolt välja toodud probleemi, milleks on vähene või puudulik arendavate mängude kättesaadavus eesti keele ruumis. Projekt on kinnise lähtekoodiga.

Arendustöö tulemil koostatakse veebirakendus, kus on võimalik registreeruda ja navigeerida erineva raskusastmega mängude vahel. Läbi tehtud mängude eest on võimalik koguda punkte, mille abil on saab avada uusi mängude tasemeid.

Töö koosneb osadest, milles koostatakse keskkonna veebiteenus ja klientrakendus ning käsitletakse nende paigaldamist serverisse koos automatiseerimisega.

Oluline osa projektis on serveri ja kliendi eraldatus, et edasiselt oleks lihtsam arendada teisi kliente veebiteenusele. Lõputöö hõlmab kasutatuid raamistikke, teeke, tehnoloogiaid ning arhitektuure, mille tulemuseks on toimiv veebirakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 5 peatükki, 15 joonist, 4 tabelit.

Abstract

Creating “Nutitund“ Web Application for Educational Games to Support Child Development

The aim of this bachelor's thesis is to create a functional public web environment for educational games. The environment makes it possible for preschool children, as well as first-class children, to spend time constructively both playing and developing themselves at the same time. The solution addresses the need for educational games, which is expressed by kindergarten teachers. The project is closed source.

As a result of the development work, a web application will be created. It will be possible to register, navigate between games of different difficulty levels and collect points for what is played. Obtained points will allow to unlock new difficulties and play new games on higher levels.

The work consists of separate parts in which the web service of the environment and the client are created and installed on a server with relevant automation in place for this process.

An important part of the project is the separation of the server and the client projects so that it would be easier to implement other clients for the web service. The thesis covers the used frameworks, libraries, technologies, and architecture that result in a working web application.

The thesis is in Estonian and contains 48 pages of text, 5 chapters, 15 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* – arvuti operatsioonisüsteemiga või rakendusprogrammiga määratud reeglistik, mille alusel rakendusprogramm kasutab operatsioonisüsteemi või teise rakendusprogrammi teenuseid
CDN	*Content delivery Network* – sisuedastusvõrk, kus sisust tehakse palju koopiaid ja puhverdatakse neid üle kogu võrgu
CI/CD	*Continuous Integration / Continuous Deployment* – pidev integratsioon (CI) ja pidev edastamine (CD) kehastavad tavade kogumit, mis võimaldavad rakenduste arendusmeeskondadel koodimuudatusi sagedamini ja usaldusväärsemalt edastada
CLI	*Command Line Interface* – käsurealiides mis on operatsioonisüsteemi või rakenduse kasutajaliides
CMS	*Content Management System* – tarkvara, mis haldab dokumente veebisaitide tarvis
CORS	*Cross-origin Resource Sharing* – domeenidevaheline ressursi jagamine on mehhanism mis piirab või lubab rakendusel pärida ressurssi mis asub väljaspool domeeni kus esialgne ressurss asetseb
CRUD	*Create Read Update Delete* – selle all peetakse silmas rakenduses olevaid funktsioone mis võimaldavad andmebaasi vastu teha vastavalt andmete loomis-, lugemis-, uuendus- ja kustutus operatsioone
CSR	*Client side rendering* – kliendipoolne renderdamine viitab veebilehe sisu renderdamise koormuse viimist klientseadmele
CSS/SCSS	*Cascading Style Sheets* – veebilehtede kujundamisel kasutatav märgendkeel. SCSS puhul on tegemist Syntactically Awsome Style Sheets ehk SASS laiend stiilifailiga mis on edasiarendus CSS märgendkeelest
DOM	*Document object model* – eeskiri kuidas objekte veebilehel esitada
DRY	*Don't repeat yourself* – viitab puhta tarkvara loomis praktikale, mis seisneb koodi taaskasutamises
DTO	*Data Transfer Object* – objekt, mis kannab andmeid erinevate protsesside vahel
HTML	*HyperText Markup Language* – enamlevinud kodeerimissüsteem veebidokumentide loomiseks

HTTP/S	*HyperText transport protocol* – protokoll teabe edastamiseks arvutivõrkudes. HTTPS ehk *HyperText transport protocol over SSL* puhul on tegemist teabe edastamisega krüpteeritud kujul
i18n	Internationalization – viitab rakenduse lokaliseerimisele või tõlkimisele keeltele mis toetaksid kasutaja emakeelt
IT	*Information Technology* – infotehnoloogia
JSON	JavaScript Object Notation – andmevahetus formaat
JWT	*JSON Web Token* – avatud standard (RFC 7519), mis määratleb kompaktse ja iseseisva viisi, kuidas osapoolte vahel JSON-objektina turvaliselt teavet edastada
ORM	*Object-relationl Mapping* – andmebaasi objektide-relatsioonide ja andmete struktuuri haldus läbi programmeerimiskeele
OS	*Operating system* – operatsioonisüsteem
PaaS	*Platform as a Service* – platvorm teenusena on pilvandmetöötluse tüüp, mille korral teenusepakkuja pakub klientidele platvormi, võimaldades neil ärirakendusi arendada, juhtida ja hallata, ilma et oleks vaja sellist tarkvara infrastruktuuri ehitada ja hooldada
REST, RESTful	*Representational State Transfer* – kliendi ja serveri vaheline kommunikatsiooni arhitektuur. Rakendusi mis kasutavad REST arhitektuuri kutsutakse „RESTful“ rakendusteks
SEO	*Search Engine Optimization* – veebilehe või veebisaidi nähtavuse suurendamine otsingumootori otsingutulemustes
SOAP	*Simple Object Access Protocol* – klient ja serveri kommunikatsiooniks kasutatav protokoll, millega veebiteenused vahetavad omavahel struktuurseid andmeid
SOLID	*Single-responsiblity, Open-closed, Liskov substitution, Interface Segregation & Dependency Inversion Principles* – viitab erinevatele praktikatele puhta tarkvara loomiseks
SQL	*Structured Query Language* – struktuurpäringukeel, enimkasutatav päringukeel, mida toetavad kõik klient-server keskkonnale projekteeritud relatsioonandmebaasid
SSR	*Server Side Rendering* – teenusepoolne renderdamine viitab veebilehtede renderdamise koormuse viimist serveri poolele
TLS	*Transport Layer Security* – SSL standardi edasiarendus
UI	*User Interface* – Kasutajaliidese lühend
VPS	*Virtual Private Server* – virtuaal privaatserver on server, mis on majutatud väljaspool kasutaja taristut, viidatakse ka kui pilveserveriks, mida kasutaja kasutab teenusena

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Lastele väga meeldib arvutis mängida, kuid tihtipeale on see loomult vaid meelelahutuslik ning ei kanna endaga kaasas erilist konstruktiivset aja veetmise viisi. Eelkooli lastele ning ka esmaklassi lastele, tuleb pöörata erilist tähelepanu nende individuaalsele arengule. Kirjaoskus ja matemaatika on ühed põhilisemad asjad, mida eelkooli lastele õpetada ning alati nende õppimine ei käi kerget viisi. Õppimine muutub eriti raskeks kui õpitav ise ei paku enam erilist huvi, mille kaudu laps kaotab tähelepanu ning on fokuseeritud muudele asjadele nagu näiteks mängude mängimine tahvelarvutis.

Paljudes algkoolides ning ka lasteaedades on rühmadesse eraldatud tahvelarvutid mis aitaksid muuta õpinguid rohkem huvitavamaks lastele. Arvutite kaudu saavad lapsed mänguliselt lahendada ülesandeid, hoides nende fookust õpitaval teemal. Probleem siinkohal on asjaolus, et pakutavaid rakendusi mida lapsed saaksid kasutada eesti digiruumis ning samuti ka eesti keeles, on kaunis vähe. Sellepärast tihti ei ole lastel võimalik neile eraldatud arvuteid kasutada ettenähtud viisil enesearenguks.

Käesolev lõputöö analüüsib nimetatud probleemi Eesti kontekstis ning vaatleb olemasolevaid keskkondi, mis pakuksid laste arengut toetavaid mänge. Tõstatatud probleemi lahenduseks on pakutud veebikeskkonna loomine eelkooli- ning esmaklassilastele, kus lapsed saaksid leida arengut toetavaid mänge. Et lapse tähelepanu köita ning, et lapsel oleks jätkuvalt huvitav keskkonnas viibida, on lahendatud ülesannete eest võimalik punkte koguda, avades uusi ja keerulisemaid ülesandeid.

Paljud taolised keskkonnad on tasulised ning toimivad individuaalsete litsentsi ostmisega iga seadme jaoks eraldi. Pakutud lõputöö lahendina on keskkond esialgselt tasudeta ning avatud kõigile kasutamiseks, keskkonna ülal hoidmiseks vajaliku summa kogumiseks on planeeritud reklaamitulu läbi reklaamide kuvamise veebikeskkonnas.

1.1 Ülesehitus

Käesoleva töö lahendamise käigus analüüsitakse põhjalikumalt probleemi ning selgitatakse välja meetodid kuidas probleemi lahendada ning pannakse paika projekti skoop. Vaadeldakse millised on võimalikud takistused mis võivad esineda antud töö raames lahenduse elluviimise juures ning kuidas neid ületada. Pakutakse välja IT (*Information Technology*) lahendus, mis aitaks olemasolevate lahenduste puudusi katta ning mida oleks võimalik edasi arendada.

Analüüsi käigus selgitatakse haldus- ja arendusvahendite valikuid ning võrreldakse erinevaid tehnilisi lahendusi klient- ja server tarkvara loomiseks. Autori poolt valitakse nende seast välja sobivaimad, mis vastavad kõige enam projekti olemasolevatele eeldustele ja kitsendustele. Tuuakse välja kasutajagrupid ning kirjeldatakse nendega seonduvad nõuded, mis on kirja pandud kasutajalugude kujul. Eelnevat arvesse võttes on piiritletud projekti skoop ning välja toodud nõuded töö funktsionaalsetel ning mittefunktsionaalsetel osadel.

Lahenduse koostamise käiku kirjeldatakse osadena, nende seas on välja toodud andmebaasi mudeli koostamine, kasutajaliidese loomine ning nende erinevad vaated, rakenduse serveripoolsed liidestused ja meetodid, kuidas turvalisemalt hallata kasutajate andmeid nii, et lastele loodud kasutajakontode andmeid oleks võimalik vaadelda ning hallata vaid volitatud isikutel.

Viimases peatükis tuuakse välja lahenduse testimistulemused ning uuritakse loodud lahenduse tõhusust. Tuuakse välja saavutatud eesmärgid ning vaadeldakse valminud projekti osi, samuti analüüsitakse töö võimalikke edasiarendusi ja suundi.

2 Analüüs

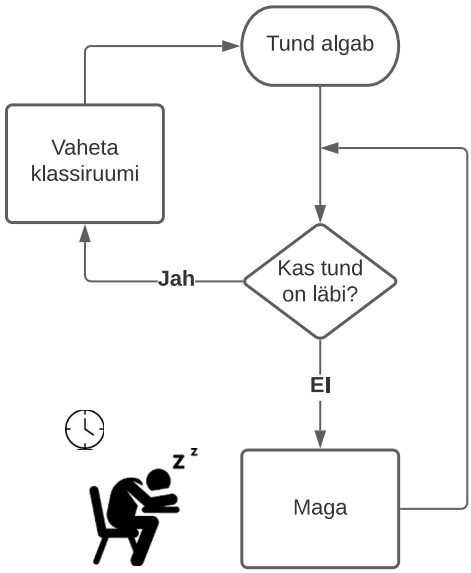
Antud peatükis analüüsib autor püstitatud probleemi tausta, võimalikke lahendusi ning antud lahenduste objektiivsust. Uuritakse funktsionaalseid ja mittefunktsionaalseid nõudeid, valitakse sobivad tehnoloogiad ning tehnoloogiline lähenemine tarkvara ehitamiseks. Seatakse projekti piiritlev skoop ning tuuakse välja antud skoobist väljaspool asuvad projekti arengusuunad. Kirjeldatakse rakenduse arhitektuuri, haldust ning majutus metoodikaid. Valitud lahendusele tehakse ärilise tasuvuse analüüs.

2.1 Taust

Meie kaasaegne IT ühiskond on täis nutimugavusi, mis aitavad meil vabal ajal lõõgastuda ning ohtralt erinevat meediat pruukida, samuti ka aidata meil teha tööd, tellida koju toiduaineid või šopata e-poes. Mugavus ning võimalused mida IT on tänapäevaks inimestele võimaldanud on suurejooneline.

Lapsele kvaliteetse hariduse võimaluse andmine on oluline, et laps saaks edukalt hakkama õpingutega koolis, töökohal ning ka üldiselt edasises elus. Siinkohas saame võtta appi IT lahendusi, et muuta ka lastele kättesaadavaks lahendusi, mis aitaksid arengut soodustada.

Laste tähelepanu hoida õpingutel on keeruline, eriti kui õpitav teema ei paku huvi, või kui materjal mida omandama peab on mahukas. Illustratsioon sellest tunnitöö raames on välja toodud Joonis 1. Lapse puuduv huvi tunnitöös. juures.



Joonis 1. Lapse puuduv huvi tunnitöös.

Kui meelelahutuseks on olemasolevaid IT lahendusi lugematu arv, on laste haridusele suunatud sisuga rakendusi nende seast arvult vähe. See nõuab sarnaste lahenduste edasiarendusi eesti keele ruumis, et muuta õpingud põnevamaks ning leida viise laste tähelepanu hoidmiseks.

2.2 Olemasolevad lahendused

Lähenedes eesti keele ruumi, on olemasolevate haridus suunitlusega rakenduste arv veelgi väiksem, olemasolevate lahenduste seast võib leida amortiseerunud või täiendust vajavaid rakendusi ja veebikeskkondi. Vaja on leida lahendus, mis köidaks lapsi rakenduses aega veetma ning mis samuti aitaks lastel arendada nende kirjaoskust või mis aitaks neil lahendada arvutamise ülesandeid.

Mitmed lahendused on kallid ning sellepärast ei ole need kättesaadavad lasteaedades ning muudes haridusasutustes. Tihti vajab iga kasutaja rakenduse kasutamiseks eraldi litsentsi, mis kokku summeeritult võib koostada liialt suure arve, et lasteaiad nende rakenduste kasutamist saaksid endale lubada.

Järgnevalt on välja toodud loetelu mõningatest populaarsematest olemasolevatest lahendustest, mis on kasutuses lastele lasteaias või mis on kasutatavad ja kättesaadavad lastele Eestis.

- „Alguse asi“ – eestikeelne animatsioonidega last hariv arvutiprogramm, mis on mõeldud pere kõige pisematele (3-7 aastastele) arvuti klaviatuuri, tähtede ja numbrite tundmaõppimiseks. Programmi kasutamine ei eelda lapselt lugemisoskust, kuna teda juhendavad ja tegevusi kommenteerivad eesti keeles kõnelevad animeeritud tegelased.

- Bee-Bot / Blue-Bot – lasteaedades kasutuses olevad robotid, mis alluvad lihtsatele liikumiskäsklustele. Harjutavad laste loogilist mõtlemist läbi robootika.

- Smartboard – Interaktiivne tahvel kuhu saab joonistada ning millel on võimalusi kujustada erinevat sisu tahvli peale.

- Matatalab - Ekraanivaba programmeerimise komplekt, mis aitab õpetada noores eas lastele programmeerimist läbi käeliste harjutuste.

- LearningApps.org – veebikeskkond, mis pakub lastele arendava sisuga mänge ning ülesandeid lahendamiseks. Šveitsi sait millel on Eesti keele tugi, tasuta kasutamiseks.

- IXL.com – Üks suuremaid Ameerika ühendriikides toimiv lastele suunatud arendavate ülesannete lahendamise veebikeskkond. Väga laialdase ülesannete valikuga, erinevad raskusastmed, erinevatele vanusegruppidele ning klassidele. Platvorm on tasuta kasutamiseks, kuid registreerumisel suunab tasulist teenust kasutama, et individuaalseid õpinguid paremini läbi viia. Eesti keele tugi puudub.

- Schoolaby.com – õppekeskkond erinevate kursuste läbiviimiseks. Keskkonnas on õpetajatel võimalus luua kursuseid, jagada materjale, pidada tunde ning organiseerida kontroll teste kursuse materjali omandamise kontrolliks. Keskkonda kasutatakse kaugõppe vormi läbiviimiseks klasside või huvigruppide seas.

2.3 Probleem ja ettepanek

Küsitledes erinevate lasteaedades töötavaid õpetajaid, on vastuseks saadud, et lastele väga meeldib IT vidinatega mängida, kuid nende sisu on tihtipeale vaid meelelahutuslik või kallis. Võrdluseks, kui kutsuda lapsi vesivärvidega joonistama, jooksevad kõik eemale

ning kaotavad huvi, kuid kui kutsuda tegema tegevusi, mis on seotud tehnikaga, on kõigil huvi suur ning silmad säravad. See sunnib kasvatajaid mõtlema välja uusi viise kuidas huvitavalt ja konstruktiivselt sisustada laste aega. Vaja oleks praktilist lahendust, mis köidaks lapsi õppima ning mis ei oleks kallis soetada või kasutada.

Sellest tulenevalt on autor välja pakkunud lahenduseks kasutada lasteaedades ja muudes haridusasutustes olemasolevaid tahvelarvuteid, et pakkuda lastele lahendada arendavaid ülesandeid ja mänge loodavas veebirakenduses, mille nimeks saab „Nutitund“. Veebikeskkonnas oleks võimalik lastel mängida arendavaid mänge või lahendada arvutamisoskust või kirjaoskust arendavaid ülesandeid, kasutades praktiliselt kõiki seadmeid, millel on olemas veebilehitseja tugi. Loodav veebirakendus annaks lapsele võimaluse seda kasutada ka oma nutitelefonis, koduarvutis või ka tahvelarvutis. Kasutatav keskkond oleks esialgselt tasuta ning kättesaadav ja kasutatav kõigi poolt, kellel on seadmega ligipääs internetile.

2.4 Skoop

Olemasolevate lahenduste üks olulisi probleeme on nende kõrge hind teenuse kasutamise eest. Vaja oleks lahendust, mis oleks odav või koguni tasuta ning mis oleks kohastatud eesti keele ruumis kasutamiseks. Antud lõputöö raames on skoobis luua veebilehitsejapõhine keskkond, mida lapsed saaksid kasutada arendavate mängude mängimiseks. Keskkonna eesmärgiks on pakkuda lastele harivat sisu olles samal ajal köitev nagu mõni populaarne meelelahutuslik mäng. Köitev välimus on eelkõige oluline et lastel ei hakkaks kiirelt igav ning et neil ei tekiks kiire soov mujal keskkonnas aega veeta. Meeldiva välimusega kasutajaliides ning lihtne lehesisene navigatsioon peavad olema tagatud kasutamiseks. Laste kasutajate andmete ligipääsetavus peab olema kontrollitud ning kasutajate info vaadeldav ainult volitatud kasutajatele.

Lähtuvalt olemasolevast ressursist peab lastel olema võimalus keskkonnas koos lahendada ülesandeid või võistelda punktide saamise eest. Lahenduse skoobis on luua lastele arendava sisuga ülesandeid ning mänge, mille hulk on esialgselt arvult väike, kuid mida saab edasiselt täiendada. Esialgseteks suundadeks on kirjaoskuse arendamine ning arvutustehted kuni kolmekohaliste arvudeni. Rakendus peab töötama nii mobiili, tahvelarvuti, kui ka töölaua vaates.

2.4.1 Turvalisus

Laste andmete turvalisuse tagamine internetis liikudes on üliolulise tähtsusega. Kui lasteaias lapsevanem usaldab oma lapse turvalisuse õpetaja kätesse, ei ole vaja lapsevanemal omakorda lisaks muretseda lapse turvalisusele, kui ta iseseisvalt lasteaias viibides navigeerib veebikeskkondades. Ka selle vastutuse peaks suunama kasvatajale ning ka arendatava keskkonna loomisel teha kindlaks, et lapse kasutajaga seotud andmete nähtavus on kontrollitud ning nähtav vaid volitatud isikutele (kasvataja, lapsevanem, rühmakaaslased).

Selleks tuleb lapse esindajal, milleks on näiteks kasvataja või õpetaja, luua konto veebikeskkonda ning samuti ka grupp alamkontosid mida anda lastele individuaalseks kasutamiseks. Sellel viisil näeb lapsega seotud andmeid vaid usaldusisik ning muudel kasutajatel puudub ligipääs nendele andmetele.

Samuti annab selline lähenemine võimaluse loodud grupi kasutajatel veebikeskkonnas lihtsasti üksteist üles leida ning võistelda või koos lahendada ülesandeid. Vanem-kasutajal on lihtsam nii alamkasutajaid hallata ning neid lisada, muuta või eemaldada vastavalt nõuetele.

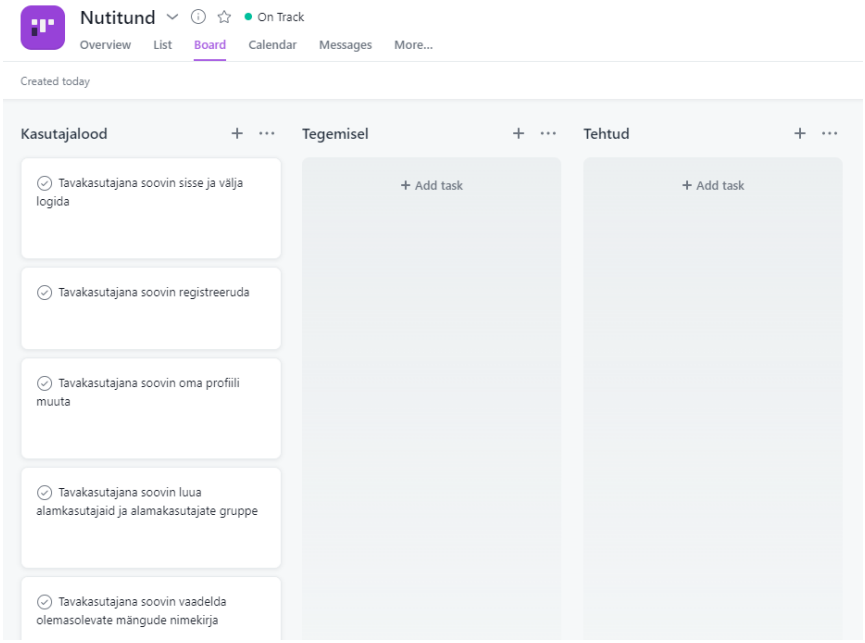
2.4.2 Kasutajalood

Selleks, et saada parem ülevaade nõuetest mida lahendus peab endas sisaldama, on autor toonud välja loetelu kasutajalugusid. Nõuete seadmisel võeti arvesse, et probleemi käsitletakse peamiselt Eesti kontekstis ja veebirakendus peab toimima kulude katmiseks piisava tasuvusega.

Nõuete koostamisel on kasutajalugudel seatud erinevad rollid. Põhirollideks on veebikeskkonna kasutaja (edaspidi tavakasutaja), õpilaskasutaja (edaspidi alamkasutaja) ja viimaks administraator. Tavakasutajaks on kasutaja, kes soovib kasutada keskkonda, lahendada ülesandeid ning samuti ka luua alamkontosid lastele kasutamiseks. Tavakasutaja peab alamkasutajate loomiseks tegema grupi, kuhu need hiljem paigutada. Alamkasutaja on sarnane tavakasutajaga rollilt, kuid erinevus on selles, et teda haldab teda loonud tavakasutaja. Alamkasutaja ei saa omakorda uusi kasutajaid luua, kuid ta näeb teisi kontosid, kes on temaga ühes kasutajate grupis. Nii tavakasutaja kui ka alam-kasutaja saavad mõlemad lahendada ülesandeid. Tavakasutajal on võimalus lahendada ülesandeid koos teiste tavakasutajatega ja tema poolt loodud grupi liikmetega, alam-

kasutaja saab seda teha vaid grupi liikmetega. Administraator haldab kõiki keskkonnas loodud kontosid, tegeleb kasutajate probleemide lahendamisega ning tõrgete eemaldamisega.

Et paremini jälgida nõuete lahendamise käiku ning mugavalt hallata kasutajalugusid, on autor kasutusele võtnud Asana veebirakenduse [1] kuhu sisestada projektiga seotud funktsionaalsed kasutajalood.



Joonis 2. Kasutajalugude haldamine rakenduses Asana.

2.4.3 Funktsionaalsed nõuded

Tavakasutaja jaoks määratletud funktsionaalsed nõuded:

- Tavakasutajana soovin sisse ja välja logida
- Tavakasutajana soovin registreeruda
- Tavakasutajana soovin oma profiili muuta
- Tavakasutajana soovin luua alamkasutajaid ja alamkasutajate gruppe
- Tavakasutajana soovin vaadelda olemasolevate mängude nimekirja
- Tavakasutajana soovin lahendada ülesandeid ja mängida mänge

- Tavakasutajana soovin ülesannete lahendamise eest saada punkte, et avada uusi ülesandeid
- Tavakasutajana soovin näha enda poolt lahendatud ülesandeid
- Tavakasutajana soovin lahendada ülesandeid koos teiste kasutajatega
- Tavakasutajana soovin näha oma grupi liikmeid
- Tavakasutajana soovin registreeruda ja sisse logida oma sotsiaalmeedia kontoga

Alamkasutaja jaoks määratletud funktsionaalsed nõuded:

- Alamkasutajana soovin sisse ja välja logida
- Alamkasutajana soovin tutvuda ülesannete ja mängude nimekirjaga
- Alamkasutajana soovin lahendatud ülesannete eest saada punkte
- Alamkasutajana soovin teenitud punktidega avada uusi ülesandeid
- Alamkasutajana soovin lahendada ülesandeid koos grupi liikmetega

Administraatori jaoks määratletud funktsionaalsed nõuded:

- Administraatorina soovin hallata kõiki veebikeskkonnas olevaid kasutajaid
- Administraatorina soovin, et veebipõhine kasutajaliides ühilduks täielikult standarditega HTML[1] 5 ja CSS[2] 3
- Administraatorina soovin, et kõikides andmebaasi tabelites on defineeritud vähemalt üks primaarvõti.
- Administraatorina soovin, et andmebaasi objektide nimetused oleksid sisulised ning annaksid aimu oma otstarbest.

[1] *HyperText Markup Language* – enimkasutatud keel dokumendimudel struktuuri kujundamiseks

[2] *Cascading Style Sheets* – enimlevinud HTML dokumendi stiili muutmise märgendkeel

- Administraatorina soovin, et klient rakendus ei pöörduks otse andmebaasi poole

2.4.4 Mittefunktsionaalsed nõuded

Tavakasutaja jaoks määratletud mittefunktsionaalsed nõuded:

- Tavakasutajana soovin, et lapsed saaksid veebikeskkonda kasutada individuaalselt
- Tavakasutajana soovin veebilehte kasutada kõigi levinud veebilehitsejatega
- Tavakasutajana soovin emakeelset kasutajaliidest, et aru saada veebilehe sisust
- Tavakasutajana soovin kasutada veebilehte mobiilil, tahvel- ja lauaarvutil
- Tavakasutajana soovin, et minu ja minuga seotud andmed oleksid kaitstud ning nende haldamine turvaline

Alamkasutaja jaoks määratletud funktsionaalsed nõuded:

- Alamkasutajana soovin, et minu kasutaja andmed on hallatud grupi looja poolt
- Alamkasutajana soovin, et minu andmed oleksid vaadeldavad vaid volitatud kasutajatele

Administraatori jaoks määratletud funktsionaalsed nõuded:

- Administraatorina soovin, et kasutajad saaksid ligi vaid neile lubatud ressurssidele
- Administraatorina soovin, et koodibaas oleks kirjutatud rahvusvahelises keeles, et saaks kaasata rahvusvahelist tööjõudu
- Administraatorina soovin, et projekti ehitamisesse oleks kaasatud pidev integratsioon ning pidev edastus (CI/CD ehk *Continuous Integration / Continuous Deployment*)
- Administraatorina soovin, et tarkvara oleks võimalik testida

2.5 Tehnoloogia valik

Meetodeid mida kasutada kaasajal tarkvarasüsteemide loomiseks on rikkalik arv [2]. Et valida sobivaim tehnoloogia projekti loomise jaoks tuleb arvesse võtta projektiga kaasnevad kitsendused ja laiendused samuti ka autoripoolne kogemus erinevate tehnoloogiate suhtes. Käesoleva lõputöö raames on fookusesse võetud vaid populaarsed või järsku populaarsust koguvad tehnoloogiad, millel on suur või järsku kasvu omav kasutajabaas.

Fookusest välja jäetud on CMS (*Content Management System*) lahendused, kuna nende kasutamine teeks arendatava lahenduse vähem paindlikuks. CMS lahendusi pigem kasutatakse kui standardlahendusi [3] ning nende kaasamine ei oleks levinud viis paindliku tarkvarasüsteemi arendamiseks. Lõputöö skoobis on luua keskkond mida saab kasutada nii nutitelefonis, tahvelarvutis, kui ka lauaarvutis. Analüüsides nõudeid mis eelnevas peatükis on välja toodud, näeme, et arendatav süsteem peaks täitma mitmeid rolle. Lahendus peab salvestama kasutaja andmeid, haldama kasutaja andmeid turvaliselt, pidama järge tegevustest mis keskkonnas tehakse, toimima erinevates klientseadmetes ning samaaegselt tegema kõike nimetatut võimalikult väikese viitega.

Sellest saab eeldada, et arendada tuleks tsentraalne, monoliitne veebiteenus või mikroteenused [4], kuhu erinevad klientseadmed saavad üheaegselt ühenduda. Kuna arendatav lahendus hinnanguliselt ei nõua individuaalsete teenusesiseste ressursside skaleeritavust, jätab autor mikroteenused siinkohal skoobist välja. Peale teenuse tuleb arendada ka klientrakendus, mis viiks läbi turvalise suhtluse teenuse ja lõppkasutaja vahel, samuti mis toimiks kõigil nõutud seadmetel.

Järgnevalt vaatleme erinevaid olemasolevaid tehnoloogiaid mis aitaksid antud lahendust edukalt koostada võttes arvesse olemasolevat ressurssi. Siinkohal soovib lõputöö autor välja tuua, et tehnoloogiate valikute tegemisel on autor arvestanud kindlate teguritega, mis tulenevad:

- Paindlikus – valitav tehnoloogia peab olema lihtsasti liidestuv ning toetama ka teisi populaarseid tehnoloogiaid. Samuti võiks kasutatav tehnoloogia olla levinud ning vähesema ajamahuga õpitav, et tulevikus oleks lihtsam kaasata projekti lisa tööjõudu.

- Teenuse ülalpidamiskulud – peamine rahastus loodaval lahendusel on reklaami pinna üürimine.

- Turvalisus – Loodav rakendus peab kaitsma kõigi kasutajate andmeid. Paljudel raamistikel on oma integreeritud viis läbi viia andmeturvet, sellest peab ennekõike olema teadlik.

- Kogemus – Ressurss on antud diplomitöö koostamisel piiratud, seda tuleb kasutada mõistlikult ning hindama ajakulu erinevate tehnoloogiate kasutusele võtmisel.

- Veakindlus ja ennetus – tehnoloogia peaks olema üheselt mõistetav. Valitaval tehnoloogial võiks olla kaasaegne puhas arhitektuur ja struktuur. Tüübikindlus, et lisada ennetavaid meetmeid pidevate pisivigade lahendamiseks, mis võivad sisse lipsata igapäevase arendustöö käigus.

2.5.1 Teenuse ühenduvus

Bakalaureusetöö nõuete kohaselt peab lahendus töötama kasutaja nutiseadmel või arvutil. See annab võimaluse kasutada seadmel olevat võrguühendust veebiteenusega ühendumiseks. Selleks, et ühendada klientrakendus veebiteenusega, peab valima sobiva arhitektuuri, mis aitaks enim saavutada nõutud funktsionaalsus.

Populaarseimad arhitektuurid andmeühenduse loomiseks teenuse ja klientrakenduse vahel on REST[1] ja SOAP[2]. REST on 2000. aastal Roy Fielding'u poolt välja pakutud tarkvaraarhitektuur interaktiivsetele veebiteenustele. Seda standardit järgivat veebiteenust nimetatakse RESTful aplikatsiooniks. Sellist arhitektuuri järgiv veebiteenus peab pakkuma oma veebiressursse tekstilises vormingus ning võimaldama neid lugeda ja muuta kasutades lubatud protokolle ja etteantud toimingute komplekti [5].

SOAP on REST eelkäija, mis pakub veidi robustsemat lahendust samale probleemile mida REST lahendab. SOAP paneb rohkem rõhku infovahetuse turvalisusele ning on rohkem standardiseeritud. Siiski võivad teenusepoolsed muudatused põhjustada vajadust

[1] *Representational State Transfer* – kliendi ja serveri vaheline kommunikatsiooni arhitektuur

[2] *Simple Object Access Protocol* – kliendi ja serveri kommunikatsiooniks kasutatav protokoll, millega veebiteenused vahetavad omavahel struktuurseid andmeid

suurtele muudatustele kliendi poolel mis omakorda võivad rakenduse halduse ja arenduse teha mahukamaks. Sisse ehitatud veahaldus ning tugi suuremate andmehulkade usaldusväärseks transpordiks võivad teha SOAP-ist ahvatleva valiku [6].

Veel on populaarsust kogumas GraphQl, mis pakub alternatiivi REST rakendustele, andmevahetuse optimeerimiseks [7]. Tegemist on andmete pärimiseks loodud keelega, mille abil luua veebiteenuseid. GraphQl ei ole arhitektuur, vaid andmekogude pärimise keel. Selle eesmärk ei ole asendada eelnimetatud tehnoloogiaid vaid pigem pakkuda lisa väärtust. Eeliseks on olla paindlikum ning võimaldada teenuse andmevahetust vähemate päringutega. GraphQl on hea valik mikroteenuste arendamisel ja kasutamisel, kuna see vähendab individuaalsete päringute tegemist mikroteenuste ja kliendi vahel.

Nimetatud tehnoloogiatest on enim leidnud populaarsust ning kasutust REST [6] ning kuna antud lahenduse raames ei ole skoobis kasutada mikroteenuseid, on ka antud lõputöö autor otsustanud võtta kasutusele REST arhitektuuri üle teiste nimetatute. Kui peaks edasiselt projekti arengus esinema nõue mikroteenustele, siis on võimalik liidestada veebiteenus GraphQl toega.

2.5.2 Veebiteenus

Ajalooliselt on IT maailmas erinevaid tehnoloogiaid ja nendega seotud tööriistu käsitletud grupeeritult. Nende gruppide eesmärk on pakkuda terviklikku lahendust omavahel hästi töötavatest vahenditest mis annavad olulist funktsionaalsust terviklike tarkvarasüsteemide ehitamiseks. Nendele grupeeringutele viidatakse kui *stack* ehk pinu. Ühed populaarsemad pinud mida tänapäeval kasutatakse on [8]:

- MEAN pinu – viitab tarkvaraarendus pinule mis koosneb MongoDB(M), Express(E), Angular(A) ja Node.js(N) tehnoloogiatest. Tarkvara pinu on avatud lähtekoodiga ning tasuta kasutatav dünaamiliste veebisaitide ja veebiteenuste ehitamiseks. Kuna kõik MEAN pinu komponendid toetavad rakendusi mis on kirjutatud JavaScriptis, on võimalik nii veebiteenust kui ka klientrakendust kirjutada kasutades ühe ja sama keelena JavaScripti.

- .NET pinu – tegemist on Microsofti poolt loodud tarkvaraarendus pinuga. Pinu kasutab peamise programmeerimiskeelena C# ning on populaarseks valikuks paljudele suurtele arendusprojektidele. Raamistik on põhjalikult

dokumenteeritud, omab tugevat andmeturvet ning turvalisust tõstvaid vahendeid, kasutab Entity Framework ORM (*object relational mapper*) lahendust andmete haldamiseks andmebaasi ja veebiteenuse vahel. Raamistik võimaldab kiirelt arendada kindla struktuuriga veebirakendusi.

- LAMP pinu – on akronüüm mis viitab tehnoloogiatele milleks on Linux(L), Apache(A), MySQL(M) ning PHP/Perl/Python(P). Pinu on olemuselt sarnane MEAN pinuga avatud lähtekoodiga, kuid mis toimib Linux operatsioonisüsteemil. Raamistik on esialgselt loodud toimima modulaarselt, et kõiki komponente oleks võimalik omavahel välja vahetada ning asendada teiste tehnoloogiatega vastavalt vajadusele. LAMP pinul on olemas ka sarnased raamistikud. Näiteks WAMP, mis töötab Windows OS-il (*operating system*) ning MAMP mis töötab MacOS keskkonnas.

- Java Spring pinu – kasutab Java programmeerimiskeelt ning Spring raamistikku rakenduste loomiseks. Antud tehnoloogia leiab laialdast kasutust suuremates tarkvarasüsteemides, kus on oluline teenuste püsiv ja kindel toimimine. Raamistik koos keelega on väga robustne ning struktureeritud tagamaks kindlate tarkvaramustrite kasutamist erinevate arendajate poolt [9].

- ROR – ehk *Ruby on Rails* on Pinu mis kasutab Ruby programmeerimiskeelt ning Rails raamistikku. Kasutatakse samuti dünaamiliste veebisaitide ja veebiteenuste ehitamiseks. Omab laia arendajate kasutajaskonda, rakendab endas arhidektuurilisi printsiipe nagu DRY (*Don't Repeat Yourself*) ja *Conventsion over configuration* [10].

- Laravel PHP pinu – üks populaarsemaid PHP keelt kasutavatest veebiteenuse raamistikest. Laravel aitab kiirelt arendada suuri tarkvarasüsteeme integreerides paljusid komponente juba raamistikku nagu kasutajate autentimissüsteem, emaili teenus või automaattestimis vahendid. Arendus kiirus ja skaleeritavus teeb selle raamistiku populaarseks valikuks paljude inseneride seas [11].

- JAM pinu – uudne lähenemine mis käsitleb endas kolme lüli, JavaScript(J), API[1](a) ja *prebuilt markup*(M) ehk eelgenereeritud veebilehe märgend. Antud pinu eesmärgiks ei ole tuua välja konkreetseid tehnoloogiaid vaid tuua välja lähenemine, kus staatilised HTML failid on eelnevalt genereeritud ning renderdatud klientmasina peal. HTML sisu hoitakse sarnastes keskkondades nagu *headless* CMS ehk ilma kasutajaliideseta sisuhaldus süsteemides. Peamisteks eelisteks selle kasutusel on odav skaleerimine kuna ei toeta teenusepoolset renderdamist (SSR ehk *Server-side Rendering*) ning saavutatakse kõrge SEO (*Search Engine Optimization*) ehk kodulehe otsingumootorite optimeerimine. SEO peamiseks eesmärgiks on tõsta loodud kodulehe leitavus läbi erinevate otsingumootorite.

Olemasolevaid tehnoloogiate pinusid leidub nimetatus veelgi, väljatoodute seast tuleb panna ka tähele, et mitmed pinud kasutavad sisuliselt samasugust lähenemist, kuid veidi teiste tehnoloogiatega. Siinkohal saab vaadelda näiteks MEAN pinu, mille populaarsed alternatiivid on MERN ja MEVN, mis vastavalt vahetavad MEAN pinus kasutatava Angular raamistiku välja kas React või Vue raamistikega.

Võttes arvesse, et uue programmeerimiskeele ning seda kasutava raamistiku õppimine võib võtta olulise osa olemasolevast lõputöö koostamise ajast, on esmalt mõistlik analüüsida raamistikke ja keeli millega autoril on eelnevalt kogemusi ja oskusi. Autori peamised kogemused ümbritsevad JavaScripti ning TypeScripti. Autoril on üldiselt rahuldavad oskused seotud C# ja Java keeltega. Raamistikest on rahuldataval tasemel kogemust .NET Core raamistikuga ning veidi ka Java Spring. Peamine professionaalne kogemus on autoril seotud klientrakenduste loomisega, kuid samuti leidub ka kogemust andmebaasi ja veebiteenuste halduses.

Käesolevas diplomitöös kasutatakse rakenduste loomisel vaid populaarsemaid viise terviklike veebirakenduste ehitamiseks. Olemasolevate pinude ning raamistike kasutamine võimaldab lühema ajaga luua struktuurset ning skaleeruvat tarkvara.

[1] *Application Programming Interface* – arvuti operatsioonisüsteemiga
või rakendusprogrammiga määratud reeglistik, mille alusel rakendusprogramm kasutab
operatsioonisüsteemi või teise rakendusprogrammi teenuseid

Järgnevalt vaatleme tabelit (Tabel 1) erinevatest raamistikest ning nendega seotud õppimiskeerukusest ja autoripoolsest kogemust.

Tabel 1. Veebiteenuste raamistike võrdlus.

Raamistik	**Keel**	**Õppimiskeerukus**	**Kogemus**
.NET Core	C#	Kõrge [12]	Keskmine
Django	Python	Keskmine [13]	Madal
Spring	Java	Keskmine [14]	Madal
Node.js	JavaScript	Madal [15]	Kõrge
Ruby on Rails	Ruby	Madal [16]	Puudub
Laravel	PHP	Madal [17]	Puudub

Vaadeldes andmeid, näeme, et väljatoodud kriteeriumite alusel sobivamateks kandidaatideks veebiteenuse ehitusel oleks .NET Core ning Node.js raamistikud. Toetudes raamistiku valikul vaid autori kogemuste pagasile, ei garanteeri antud projekti jaoks sobivamate tööriistade valikut ning selle edenemist. Lisaks tuleb analüüsida rakenduse nõudeid, et filtreerida valikuid veel enam.

Tehtav veebiteenus peab serveerima veebisisu klientrakendusele. Veebiteenus ei pea toimima mikroteenuste mudelis vaid võib toimida kui monoliitne API. Kui eesmärgiks on pakkuda kasutajatele tasuta teenuse ligipääsu, on mõistlik veebiteenuse jätkusuutlikkuse huvides ülalhoidmise kulusid vähendada nii palju kui võimalik. Kasutades CSR lähenemist veebilehtede renderdamiseks ning jättes välja SSR lähenemise skoobist. Nii on meil võimalus viia protsessimiseks vajalik ressursinõue klientrakendusele, tuues madalamale koormuse veebiteenusel, samuti ka ülalpidamiskulud. Siiski sellega võib tekkida veebikeskkonna jaoks kehvemad SEO võimalused ning samuti peab panema rohkem tähelepanu turvalisuse kvaliteedi hoidmisele [18].

Praktiliselt kõigi välja toodud raamistikega (Tabel 1. Veebiteenuste raamistike võrdlus on võimalik saavutada CSR valmidus, peamine erinevus seisneb keeles ning

õppimiskõvera sügavuses. Kuna projekti on plaanis edasiselt kaasata tarkvarainsenere ja selle õnnestumisel teha palju edasiarenguid, on hea valida raamistik mis on tugevalt tüübitud, kasutab puhta koodi printsiipe nagu SOLID[1] või DRY mis on kindla struktuuri ja loogilise ülesehitusega. Võttes arvesse nimetatud kriteeriume, on valitavateks raamistikeks .NET Core või Spring.

Viimastel aastatel on populaarsust kogunud NestJs raamistik, mis baseerub Node.js peal. NestJs toob Samasuguse arhitektuurilise struktuuri mis leidub .NET Core ja Spring raamistikes, kasutades TypeScripti peamise keelena. Erinevalt teistest, baseerub Node.js *single threaded non-blocking* arhitektuuril, mis sisuliselt tähendab, et veebiteenus saab pidevalt teenindada uusi ühendunuid kliente, ilma, et peaks eelmise kliendi transaktsiooni lõpetama. Selline lähenemine aitab tõsta jõudlust veebiteenustele, mis ei tee raskemat protsessimist (näiteks suurte andmehulkade renderdamine, andmekogude protsessimine) olles samal ajal kergekaaluline ja tõhus [19] [20]. Arendatava veebiteenuse raames ei ole nõuet SSR või muu raskemat sorti protsessi jooksutamiseks veebiteenusel mispärast NestJs oleks siinkohal sobilik. Samuti on ka autoril enim kogemust JavaScript/TypeScript programmeerimiskeelega, mis aitaks kiirendada arendatava keskkonna loomist. Järelduseks on autor otsustanud kasutada veebiteenuse ehitamiseks NestJs raamistikku.

NestJs raamistiku õppimiskõver võib olla järsk arendajatele kes on harjunud kasutama lõdvemalt tüübitud programmeerimiskeeli nagu Python või PHP [21] [22]. Kuna autor on kursis sarnaste arhitektuuriliste struktuuridega mis leidub .NET Core või Spring raamistikes, ei ole ka õppimiskõver järsk.

2.5.3 Klient

Võimalusi arendada klientrakendust on tänapäeval palju. Kuna lõputöö raames on nõuete kohaselt vaja luua veebiteenusele klient, tuleb uurida lähemalt millised on sobivamad veebikliendi loomiseks olemasolevad vabavaralised lahendused. Autor võtab valikute sekka vaid populaarsemad raamistikud.

- React.js – raamistik mida esitletakse kasutajaliideste loomiseks. React.js on loodud Facebook-i poolt 2013 aastal ning on leidnud suurt populaarsust paljudes

[1] *Single-responsiblity, Open-closed, Liskov Substitution, Interface Segregation & Dependency Inversion Principles* – viitab erinevatele praktikatele puhta tarkvara loomiseks

kliendipoolsetes projektides. React.js vajab paljusid kolmandate osapoolte teeke, et terviklikult toimida projektis. Õppimiskõver on keskmine ning struktuuri poolest kasutatakse kasutajaliidese elementide eraldamiseks komponente, tagamaks head skaleeritavust [23].

- Angular – TypeScripti põhine tugevalt struktureeritud raamistik. Loodud Google poolt 2010 aastal. Angular-i eelis on selle heade arenduspraktikate juurutamine projekti mis annab loodavale rakendusel võimaluse skaleeruda ning samal ajal hoida projekti hallatavana. Angular-il on suur õppimiskõver, mis võib esialgselt nõuda palju aega arendajal raamistiku õppimiseks. Angular sisaldab endas paljusid teeke, mida on võimalik tervikliku veebirakenduse arendamiseks kasutada, see omakorda tõstab raamistiku faili suurust mida kliendid peavad alati alla laadima veebilehe külastamiseks [24].

- Vue.js – progressiivne veebiraamistik suuremate veebirakenduste ja veebisaitide ehitamiseks. Vue.js on eelnevast kahest palju noorem, loodud endise Google töötaja Evan You poolt, pakub madalamat õppimiskõverat hoides samal ajal koodibaasi struktureeritud. Vastavalt Github-i repositooriumi tähtedele, on Vue.js antud lõputöö kirjutamise hetkel populaarseim raamistik maailmas [25].

- jQuery – loodud 2006 aastal, on jQuery üks varasematest suurt populaarsust omavatest klientraamistikest, mis avalikuses populaarseks sai. jQuery ei ole sarnane raamistik nagu eelmised kolm, vaid teek mille abil paindlikumalt hallata veebilehe DOM (*Document Object Model*) elemente. Samuti aitab see lihtsamalt hallata HTTP (*HyperText Transport Protocol*) päringuid. Aastatega on jQuery poolt pakutud funktsionaalsus sisse ehitatud standard veebiarendus keeltesse, mis vähendab antud raamistiku kasutamise vajaduse uutes projektides [24].

Vaadeldud raamistikest jätab autor skoobist välja jQuery, kuna nimetatu ei ole raamistik vaid pigem pakett koodi optimeerimiseks. Ühegi raamistiku puhul ei ole rahalisi takistusi kasutamiseks, samuti on kõik vabavaralised.

Antud lõputöö autor on valinud projektis kasutamiseks Vue.js raamistiku, kuna see pakub suurt paindlikkust komponentide koostamisel, omab struktuuri koos tüübitud keele toega, milleks on TypeScript. Omab madalamat õppimiskõverat ning väiksemat paketi suurust. Vue.js on samuti kiire ning omab suurt populaarsust [23].

Veebilehe stiliseerimiseks on võimalik kasutada HTML ja CSS. Teha kõik stiil kasutades vaid CSS stiilifaile võib olla keeruline ja aeganõudev. Selleks, et kiiremalt ning efektiivselt stiliseerida veebilehte on mõistlik kasutusele võtta mõni CSS raamistik. Järgnevalt vaatleme populaarsemaid CSS raamistikke, et selgitada nende sobivust:

- Semantic UI – kiiret populaarsust koguv raamistik, mis pakub paindlikkust ja lihtsust. Mis teeb Semantic UI eriliseks, on selle omadus kombineerida keeruline stiilikood koos loomuliku inimkeelega, muutes koodi iseenesestmõistetavaks. Ei ole sobilik algajatele arendajatele, kellel puuduvad baasteadmised JavaScript-ist ja veebiarendus metoodikatest [26].

- Foundation – mõeldud reageerivate ja kiirelt toimivate veebisaitide arendamiseks. On keerulise ülesehitusega, kuid pakub tuge ja struktuuri suurte veebisaitide jaoks. Raamistiku kasutavatest inimestest koosnev kogukond on üsna väike, mis võib probleemide tekkimisel raskendada täiendava informatsiooni ja lahenduste leidmist [27].

- Bootstrap – Kõige enam tuntust omav CSS raamistik. Bootstrap-i kasutajaskond on suur ning olemasolevaid koodinäiteid on palju. Sobib hästi ka algajatele arendajatele, kuna kood on lihtsasti arusaadav ning dokumentatsioon on hästi defineeritud [28].

Sarnaselt klientrakendus raamistikega, ei sea ka nimetatud CSS raamistikud takistusi lõputöö nõuete täitmiseks. Rakendused on vabavaralised ning loodud tasuta kasutamiseks. Autoril puuduvad erilised kogemused CSS raamistike kasutamisega ning kuna oluline on ajalise ressursi efektiivne kasutamine on autor otsustanud kasutada Bootstrap raamistikku. See aitab veebilehe struktuuri stiliseerimist muuta mugavamaks ja paindlikumaks.

2.5.4 Andmebaas

Peamisteks andmebaasi tüüpideks on olemas relatsioonilised andmebaasid (SQL[1] andmebaasid) ja mitterelatsioonilised andmebaasid (NoSQL[2] andmebaasid). On olemas veel eri tüüpi andmebaase, kuid nimetatud pakuvad siinkohal suuremat väärtust. NoSQL andmebaasid on peamiselt head suurte andmete hoiustamiseks, kus andmetel puuduvad üksteisega erilised relatsioonid, andmeid talletatakse mitut moodi vastavalt andmebaasi võimekustele. Levinumad NoSQL andmebaaside andmete talletamisvormingud on võti-väärtus paaridena ja dokumentidena, kus väärtusena suudab andmebaas talletada keerulisemaid andmevorme. Relatsioonilise andmebaasiga on lugu teistpidi. Loodavas projektis on palju andmeid millel on oluliselt relatsioone andmetega, seega jätab autor skoobist välja dokumendi andmebaasid ning võtab vaatluse alla vaid relatsioonilased andmebaasid. Järgnevalt vaatleme mõningaid populaarsemaid relatsioonilisi andmebaase mis oleksid sobivad kandidaadid antud projektis kasutamiseks [29].

- PostgreSQL – Berkeley ülikoolis Californias arendatud Postgre andmebaasi vabavaraline andmebaas. On populaarset kasutust leidnud kuna pakub andmete haldamiseks tugevat struktuuri, terviklikkust ning taastamise ja töökindluse nõudeid. Üldiselt teeb PostgreSQL lihtsamaid operatsioone ebaefektiivselt, mispärast valitakse tihti MySql selle asemel [30].

- MySQL – vabavaraline, lihtne ja kerge õppida. Omab mitmeid turvalisusega seotud aspekte. Vajab litsentsi, kui kasutada kommertsaplikatsioonides. On väga paindlik ja skaleeruv andmebaas [31].

- MariaDB – on loodud endiste MySql arendajate poolt, mõeldud asendama MySQL andmebaasi. MariaDB on vabavaraline ning avatud lähtekoodiga. Omab praktiliselt kõiki omadusi mis MySql, kuid on tasuta kasutamiseks ning ei vaja litsentsi ka kommertsrakendustes kasutamisel [32].

- Oracle - relatsiooniline andmebaas, mis on kirjutatud C++ keeles. Oracle andmebaasid on ühed võimekamad maailmas. Oracle andmebaasi kasutus maksab

[1] *Structured Query Language* – struktuurpäringukeel, enimkasutatav päringukeel, mida toetavad kõik klient-server keskkonnale projekteeritud relatsioonandmebaasid

[2] Sarnane SQL keelele, kuid funktsioonilt konkreetselt mitterelatiivsete andmebaaside halduseks

raha, täisteenusena on see hea valik millel on suurepärane klienditeenindus ja usaldusväärsus. Oracle on välja andnud ka Oracle NoSQL andmebaasi, et olla klientidele võimalikult paindlik [33].

MySQL oma omaduste poolest on suurepärane valik loodava projekti sees kasutamiseks, kuid kuna projekti lähtekoodi ei ole plaanis avalikustada, tuleb osta litsents MySQL andmebaasi edaspidiseks kasutamiseks. Autor otsustanud kasutada MariaDB andmebaasi, kuna antud andmebaas on tasuta kasutamiseks, vabavaraline ning ei nõua litsentsi kommertsrakendustes kasutamiseks. MariaDB omab samu omadusi mis MySQL, seega on paindlikkus ja kiirus tagatud andmehulkade haldusel.

2.6 Versioonihaldus keskkond

Projekti koodibaasi loomiseks on vaja keskkonda kuhu kood hoiustada ning vahendeid kuidas seda hallata. Tänapäeva koodihaldus keskkonnad omavad palju lisa tööriistu, mis teevad projektikoodi haldamise mugavamaks ja paindlikumaks. Järgnevalt vaatame enimkasutatud versioonihaldus keskkondi projektikoodi hoiustamiseks [34]:

- Github – Üks populaarsemaid keskkondi projektikoodi hoiustamiseks ja haldamiseks. Omanikuks on Microsoft. Omab tasuta võimalusi kasutamiseks, kuid koodi repositooriume saab ilma tasudeta luua vaid avatud lähtekoodiga projektide jaoks. Keskkonnal on tugevad integratsioonivõimalused mitmete teenusepakkujatega mis teeb kasutamise paindlikumaks.

- BitBucket – Omanikuks on Atlassian. Bitbucketi eripäraks on tema tugev liidestus ettevõtte poolt pakutud teiste populaarsete tarkvaraloomis tööriistadega nagu näiteks Jira. Omab võimalusi tasuta privaatsete koodirepositooriumite loomiseks kuni 5 liikmelistele meeskondadele.

- GitLab – Erinevalt teiste nimetatud keskkondadega on Gitlab vabavaraline ja avatud lähtekoodiga. Gitlab omab väga palju võimalusi tasuta privaatse koodibaasiga projektide arendamiseks, kus meeskonna suurus ei mängi rolli. Gitlab pakub ka võimalust jooksutada tervet keskkonda oma enda serveri peal.

Kõik keskkonnad toetavad erinevaid tööriistu pideva integratsiooni ja pideva edastuse (CI/CD) täitmiseks. Loodava projekti lähtekoodi ei ole plaanis avalikustada täiel määral,

mistõttu GitLab tundub parim variant millega edasi minna. Autor on otsustanud kasutada siinkohal GitLab keskkonda, kuna see pakub tasuta võimalusi privaatse koodibaasi loomiseks, samuti pakub väga palju võimalusi tasuta projektide arenduse läbiviimiseks. GitLab pakub võimalust paremaks CI/CD läbiviimiseks integreerides oma enda arvuti või serveri ressurss, et jooksutada kiiremini tegumijooksutajaid (*task runner*). Antud jooksutajaid saaks näiteks panna toimima olemasoleva serveri Docker konteinerisse. GitLab võimaldab ka terve nende poolt pakutava keskkonna jooksutamist oma serveril ilma tasudeta, see annaks lisa koormust teenuse administreerimise jaoks, kuid ka paindlikkust kasutada olemasolevat ressurssi lisaraha maksmata. See on suur pluss kiiremaks projektifailide ehitamiseks ning edastamiseks haldusteenusele, kust rakendust välja serveeritakse [35].

2.7 Süsteemi haldus

Selleks, et loodud rakendust oleks klientide poolt võimalik vaadelda, on tarvis serverit. Selleks, et teenust välja serveerida klientidele on kaasajal võimalik võtta palju erinevaid lähenemisi.

On võimalus teha klassikaline lähenemine, luua server ning paigaldada selle peale teenused mis jooksutaksid vajalikke funktsioone rakenduse välja serveerimiseks. Selline lähenemine võtab palju aega ja ressurssi üles seadmiseks ja ka haldamiseks. Selle lähenemise puhul peab ka pidevalt administreerima ja uuendama paigaldatud teenuseid, et need jätkuvalt toimiksid korrektselt ja ei satuks turvariskidest tulenevate ohtude ohvriks.

Teine lähenemine, mis on üks populaarsemaid tänapäeval, on kasutada pilveteenust rakenduse serveerimise keskkonnana. Sellistele teenustele viidatakse kui PaaS (*Platform as a Service*) ning nende kasutamine on väga mugav ja paindlik. Paas tüüpi teenuse administreerimist ja auditeerimist viib läbi ettevõte kes teenust osutab, mis annab vabad käed kliendil töötada vaid oma rakenduse haldamisega. Negatiivne külg on asjaolu, et nende teenuste kasutamine on reeglina kallis, ja kui kasutavate klientide arv tõuseb, mitmekordistuvad suuremast ressursivajadusest tulenevad teenuse arved. Ühed populaarsemad PaaS tüüpi serverkeskkonnad rakenduste serveerimiseks on Google App Engine, Heroku ja Digital Ocean App Platform [36] [37].

Alternatiiviks eelmisele on võimalik ka ise jooksutada PaaS tüüpi rakendusserverit oma enda infrastruktuuri peal. See on kui kompromiss administratsiooni ja rahakulu vahel. See lähenemine on üldjuhul tasuta, kuid vajab siiski detailseid teadmisi serverite administratsiooni valdkonnast. On ka olemas ise majutatavad Paas keskkondi millel on sisse ehitatud rakendused mis teevad administratsiooni ja halduse kergemaks isikutele kellel on soov lihtsalt serveerida oma ehitatud teenuseid. Nendeks on näiteks Rancher ja CapRover mis utiliseerivad vastavalt Kubernetes ja Docker Swarm tehnoloogiaid oma funktsionaalsuse realiseerimiseks [38].

Võttes arvesse, et loodav keskkond „Nutitund“ on planeeritavalt toimiv peamiselt vaid reklaamitulul, on mõistlik kasutusele võtta server lahendus mis ei nõua palju rahalist ressurssi. Kuna autor on ka eelnevalt tegelenud enda poolt majutatud PaaS süsteemidega ja jooksutanud nendel üheaegselt mitmeid erinevaid teenuseid eriliste probleemideta, on autor otsustanud edasi minna ise majutatud PaaS süsteemiga. Erinevalt Rancher süsteemist, on CapRover palju lihtsustatud kasutusega, toetamaks arendajaid kelle peamine ekspertiis ei ole süsteemi administratsioon.

On ka oluline panna tähele, et taoliste süsteemide jooksutamine oma enda serveri peal ei maksa raha, maksab siiski raha server ise, mille peal teenust jooksutatakse. Selle jaoks on võimalus hankida eraldiseisev füüsiline server arvuti või saab ka rentida kuutasu eest VPS (*Virtual Private Server*) teenust mis asendaks füüsilise serveri olemasolu vajadust. Viimase kasuks on autor otsustanud, kuna siis ei pea panema lisa ressurssi füüsilise serveri halduse peale ning serveri töös hoidmine vajab niisamagi tasu kasutatava elektri eest.

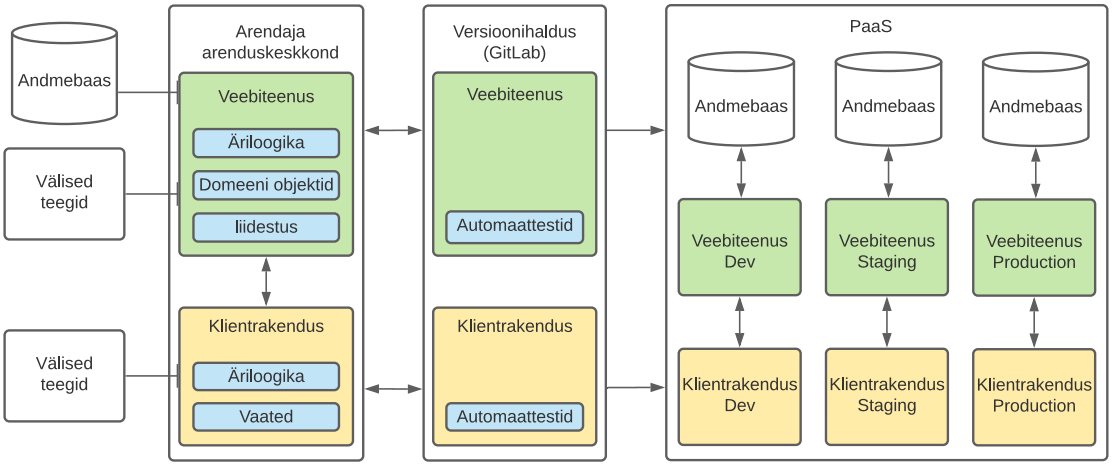
2.8 Süsteemi arhitektuur

Keskkonda koostavateks komponentideks on:

- Klientrakendus
- Veebiteenus
- Andmebaas
- GitLab tegumijooksutajad

Versioonihalduse osas on autor GitLab instantsi jooksutamise asemel võtnud kasutusele tegumijooksutajate jooksutamise PaaS keskkonnas. Nii on võimalik vähem koormust panna serverile, hoides samas kõrgemat jõudlust CI/CD töös.

Järgnevalt on välja toodud veebikeskkonna arhitektuuriskeem, et paremini mõista erinevaid projekti komponente ja nende toimimist üksteisega (Joonis 3):



Joonis 3. Veebirakenduse arhitektuur.

Nii klientrakendust kui veebiteenust arendatakse eraldiseisvatena. Neil mõlemal on oma versioonihaldus, liidestus väliste teekidega, automaattestimine ja keskkonnad rakenduse käitamiseks. Keskkonnad mida mõlemad kasutavad on *development environment* ehk arendus keskkond, mis on mõeldud konkreetselt arendajatele ja testijatele koodi testimiseks. Peale selle on *staging* keskkond mille eesmärk on näidata tarkvara funktsionaalsusi, teha viimaseid kontrolle, ennem kui see avalikustatakse. Viimasena *production environment*, kust loodud rakendused reaalselt avalikustatakse lõppkasutajatele kättesaadavaks. Oluline on ka tähele panna, et iga keskkonna jaoks on rakendusel oma andmebaas, et eristada testandmeid pärisandmetest, samuti, et läbi viia uusi arendusi nii, et ühe keskkonna andmebaasi muudatused ei mõjutaks teisi keskkondi.

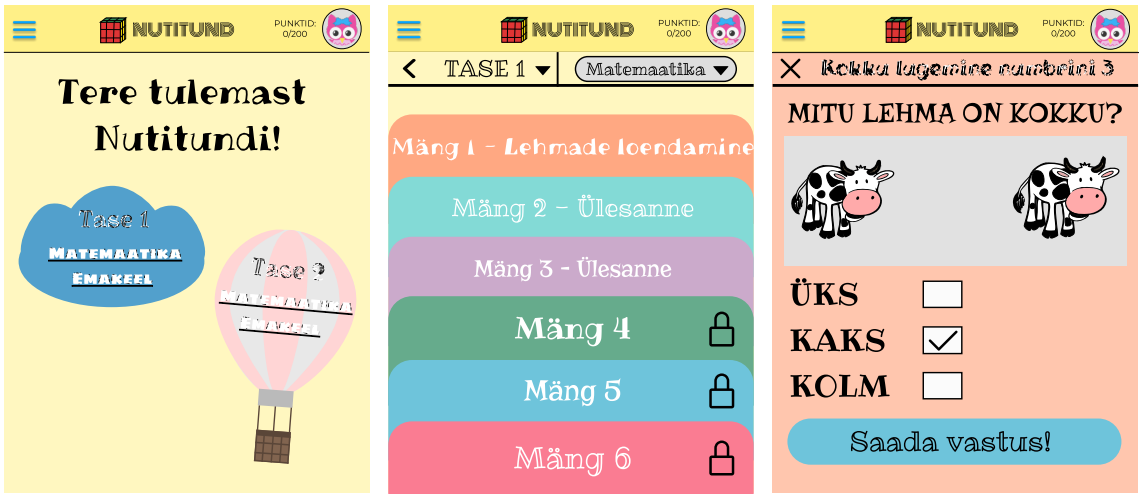
2.9 Disain

Antud peatükis vaatleme milline näeb välja kasutajaliides lõppkasutaja jaoks. Katame kasutajakogemusega seonduvad küsimused ning kirjeldame andmebaasimudeli disaini.

2.9.1 Kasutajakogemus ja kasutajaliides

Ennem kasutajaliidese ehitamist on mõistlik teha kavand, et paremini saada aimu loodava liidese sobivusest terviklikult. Selleks on autor kasutanud populaarset prototüüpimis[1] tarkvara mille nimeks on Figma. Luua tuleb kasutajaliidesed mobiili ja töölaua vaatele. Kuna tahvelarvuti kuvasuhe ja töölaua kuvasuhe on orienteeruvalt sarnased, piirdub autor esialgselt kahe vaate loomisega.

Mobiili kasutades on võimalik ekraan keerata kummuli ehk maastiku vaatesse (*landscape*), nii on võimalik saada laiem, kuid madalam ekraanikuvasuhe. Autor jätab maastiku vaate toe skoobist välja ning pühendab tähelepanu vaid mobiili püstipidi ehk portree (*portrait)* vaatele ning töölaua vaatele. Järgnevalt saame näha milline näeb välja *wireframe*[2] disain mobiili vaate jaoks (Joonis 4):



(a) (b) (c)

Joonis 4. Nutitund mobiili vaade: (a) pealeht, (b) mängude ja tasemete nimistu, (c) mängu vaade.

Veebilehel on kolm peamist vaadet mille vahel kasutaja põhiliselt navigeerib: pealeht, mängude ja tasemete loend ning viimaks mängu vaade ise. On ka eraldi kasutaja konto lehekülg, kus kasutajal on võimalus hallata kõiki kontoga seotud toiminguid ning ka lehekülg, kus kasutajal on võimalus luua gruppe ning uusi alamkasutajaid nende haldamiseks.

[1] Prototüüpimise all peetakse silmas loodava rakenduse interaktiise maketi loomist

[2] Wireframe all viidatakse veebisaidi interaktiivse maketi struktuurile

Töölaua vaadetes on sisuliselt samad vaated nagu mobiilil, kuid väikeste muudatustega, et paremini kohenduda suurema ekraaniga.

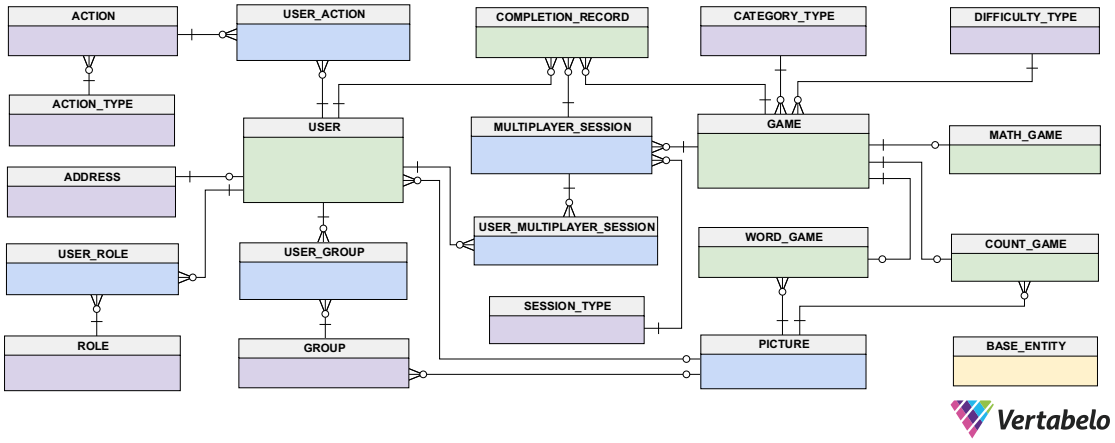
Selleks, et sisse logitud kasutaja saaks luua oma õpilastele alamkasutajaid, ei tohi ta ise olla alamkasutaja staatuses. Järgnevalt on välja toodud vaade, kus kasutajal on võimalus luua gruppe kuhu alamkasutajaid paigutada (Joonis 5).

Joonis 5. Kasutajagruppide ja alamkasutajate haldamise vaade.

Alamkasutajate gruppide loomisel on samuti võimalus samas vaates luua uus kasutaja mis mugavalt otse gruppi lisada. Hiljem on lapsel võimalus keskkonda sisse logida kasutades sisselogimisvormi grupikasutajate jaoks. Seal tuleb täita kolm vormivälja, esimene on grupi nimi, millega alamkasutaja on seotud, teiseks kasutajatunnus ning viimaks parool.

2.9.2 Andmebaasi disain

Võttes arvesse keskkonna kirjeldatud nõudeid läbi kasutajalugude ning analüüsides koostatud kasutajakogemuse disaini, on välja kujunenud olemi-suhte diagramm. Diagramm ise on loodud kasutades andmebaasi modelleerimis tööriista, mille nimeks on Vertabelo. Kuna diagramm on mahukas on seda võimalk täie detailsuse juures vaadelda peatükis LISA 2. Diagrammi lihtsustatud kuju on leitav all välja tooduna Joonis 6 juures.



Joonis 6. Olemi-suhte diagramm lihtsustatud kujul.

Olemi-suhte diagrammi koostamisel on osad tabelid markeeritud erinevat värvi, et anda paremini mõista nende rolli skeemis. Rohelist värvi tabelitega on tegemist primaarsete tabelitega, mille kasutamine on suures jaos seotud keskkonna põhifunktsioonide toimimises. Sinist värvi tabelite juures on tegemist sekundaarsete olemitega, kelle olemus on toetav põhifunktsioonide täitmiseks keskkonnas. Viimaks on lillat värvi tabelid, mille funktsioon on pakkuda kirjeldavat sisu, nagu kasutaja tegevuste logi või kontoga toimunud sündmused. Skeemis on kollase tabelina välja veel toodud baastabel, mis on laiendina kasutatud kõikide andmebaasi tabelite juures.

2.10 Ärilise tasuvuse analüüs

Rakendus koostatakse esialgselt tasuta kasutamiseks kasutajatele. See tähendab, et tulud rakenduse ülal hoiuks tuleb leida muudest allikatest, kui tasud teenuse kasutamise eest. Selleks, et veebirakendus toimiks jätkusuutlikult on minimaalne orienteeruv vajalik summa kulude katteks 100 eurot aastas. Selle eest on võimalik tasuda veebikeskkonna domeeni kulud ning veebiteenuse jooksutamiseks vajalikud serveri kulud.

Tulu teenimiseks on mitmeid alternatiivseid meetodeid. Järgnevalt toob välja autor mõningat võimalikud viisid [39] :

- Reklaamide kuvamine veebisaidil – Tõenäoliselt kõige tasuvam viis rakenduse finantseerimiseks. Siinkohal saab kuvada suunatud reklaame kasutajatele erinevatest allikatest nagu Google Adsense, Facebook Ads või Instagram Ads. Silmas peab pidama, et reklaamid oleksid suunatud lastele.

- Tasuta ja maksustatud kasutustasemed – Idee on luua uutele kasutajatele võimalus tasuta kasutada teenust, kuid see on piiratud. Kui kasutaja soovib avada teenuse lisavõimalusi või paremat sisu, tuleb tal tasuda selle eest.

- Füüsiliste meenete müümine – Kui teenus toimib mõne populaarse organisatsiooni või ettevõtte jaoks, võivad nad müüa seonduva sisuga meeneid, näiteks mütse, kampsuneid, särke. Sarnaneb viisile kuidas kinod teenivad kasu popkorni ning karastusjoogi müügist.

- Sponsoreeritud sisu – Partnerluse abil on võimalik saavutada rahaline sissetulek, kui kuvada partner ettevõtte poolt soovitud tooteid või sisu. Nii tehakse reklaami partneri soovitud sisule ning on ka võimalik peale sissetuleku tõsta teenuse mainet, kui nähakse tuntud või populaarse ettevõtte partnerlust.

- Annetused – Võimalik sissetuleku allikas, kuid see ei ole alati kindel alus millele toetuda, et hoida teenust püsivalt töös.

Peale nimetatu leidub veel erinevaid võimalusi rahastamiseks, kuid lähtuvalt teenuse sihtgrupist, jääme nimetatu juurde. Kuna teenuse peamisteks kasutajateks on lapsed, võib loota, et neid ei sega niivõrd lehe äärises olevad reklaamid, kui täiskasvanud inimesi. Seda arvesse võttes, otsustas autor esialgselt pakkuda reklaamipinda veebilehel. Edasiselt võib arendada partnerlust ettevõtetega või kontakteeruda erinevate haridust koordineerivate asutustega, et arutada rahastus võimalusi ning võimalusi viia rakendus laiemale kasutajaskonnale koolidesse või lasteaedadesse.

2.11 Analüüsi kokkuvõte

Analüüsis käsitleti tehnoloogiaid ja lähenemisi mis on olulised lõputöös seatud väljundite realisatsiooniks. Võeti kokku ülesande püstituses seatud nõuded ning vaadeldi parimaid metoodilisi ja tehnoloogilisi valikuid nõuete täitmiseks.

Analüüsi tulemis ilmnes, et tehnoloogiliselt on vaja luua veebikeskkond mis koosneb veebiteenusest ja klientrakendusest. Veebiteenuses peab sisalduma äriloogika mis läbi REST päringute suhtleb klientrakendusega kasutades HTTP/S protokolli.

Klientrakendus peab toimima isoleeritult veebiteenusega, nii on võimalik tulevikus teha edasiarendusi kliendis, ilma, et muudatused mõjutaksid veebiteenust. Järgides sellist lähenemist on soovi korral lihtne teha edasiarendusi IOS või Android rakenduste arendamisel veebiteenuse jaoks. Klientrakenduse arendamisel kasutatavaks raamistikuks osutus Vue.js koos Bootstrap CSS raamistikuga. Nii klient- kui serverrakendus kasutavad arenduseks sama keelt, milleks on JavaScript ja TypeScript. Autoril on palju kogemust antud keelega ning samuti on see üks populaarsematest keeltest mis leiab kasutust tarkvaraarendusel.

Veebiteenuse kasutatavateks tehnoloogiateks osutusid NestJs koos MariaDB andmebaasiga. NestJs toob sarnase struktuuri ja parimad tarkvara arendus praktikad nagu on kasutusel .NET või Java Spring raamistikes, kuid vähendades nende raamistike kasutamisega seotud keerukusi. NestJs toimib NodeJs tehnoloogial, mis annab sarnase ülesehitusega veebikeskkondadele nagu Nutitund tugeva jõudluse ja optimeerituse. Andmebaasi lahenduseks on valitud MariaDB, kuna on üks kiiremaid ja kergemaid relatsioonilisi andmebaase millel on suur kasutajaskond tänu selle sidemetega MySQL andmebaasiga. MariaDB on vabavaraline ja mõeldud tasuta kasutamiseks ka kommertsrakendustes, mispärast on see suurepärane valik kasutamiseks siin projektis.

Loodud rakenduste välja serveerimiseks lõppkasutajatele on võetud kasutusele PaaS lähenemine, mis koos olemasoleva infrastruktuuril jooksutamisega pakub kõrget jõudlust hoides samal ajal teenustasud madalad. Peale selle on autor kasutusele võtnud GitLab tegumijooksutajad, mida jooksutada samas PaaS keskkonnas. Selle eesmärgiks on ilma lisa teenusetasudeta saavutada kõrgem CI/CD jõudlus rakenduste ehitamiseks ja edastamiseks arendusprotsessi vältel. GitLab versioonihaldus keskkonna kasutamine annab võimaluse luua privaatseid koodi repositooriume ning annab võimaluse soovi korral lisada uusi meeskonnaliikmeid projekti arendamiseks, ilma lisa tasudeta.

3 Lahenduse arendus

Siinkohal toob autor välja veebikeskkonna arendust hõlmavad osad. Järgnev on jaotatud nelja peamisesse peatükki. Esmalt toob autor välja veebiteenusepoolse ehk serveripoolsega seonduva, teisalt klientrakenduse arendusega seonduva, kolmandaks vaatleme rakenduse testimisvõimalusi ning viimaks kirjeldab autor lahenduse integratsiooni majutuskeskkonda paigaldamiseks.

3.1 Veebiteenus

Rakenduse veebiteenus on ehitatud NestJs raamistiku peal. NestJs omab endas mitmeid olulisi funktsioone mis aitavad pakkuda kasutajale turvalist keskkonda andmete pärimiseks ja saatmiseks. Veebiteenus hoiab endas ühendust andmebaasiga, kuhu on talletatud kõik klient rakenduse jaoks vaja mineva äriloogikaga seonduv informatsioon. Järgnevates peatükkides toob autor välja veebiteenuse arendusega seotud konkreetsused.

3.1.1 Veebiteenuse arhitektuur

NestJs kasutab *single threaded non-blocking* arhitektuuri mis aitab teenusel olla madala ressursi nõudega ning samal ajal teenindada mitut klienti korraga. Raamistik ise kasutab põhjana Express.js alamraamistikku. Selle abil loob NestJs kontrollereid teenuse ressursside kättesaadavuse jaoks ning haldab kliendi ja teenusevahelisi ühendusi [20]. Üks oluline tähelepanek antud raamistiku arhitektuuri juures on selle omadus serveerida üheaegselt mitmeid klient samal ajal utiliseerides vaid ühte protsessori tuuma. Siiski koormuse tõustes on vaja ka rohkem riistvara ressurssi veebiteenuse sujuvaks toimimiseks [19].

Selleks, et NestJs projekt luua saab kasutada nende poolt pakutud NestJs CLI (*Command Line Interface*) tööriista, mis genereerib projekti põhja kus on baas teegid ning baas teenuse kontrollerid testimiseks. Genereeritud projektis on automaatselt loodud näidis sisu mida saab käivitada nende poolt etteantud käsuga „npm start“. Nimetatud käsu sisestamisel projektikaustas läbi käsurealiidese, käivitab NestJs näidisrakenduse ning serveerib ühendust kasutamiseks „localhost:3000“ aadressil [40].

Järgnevalt on tabel failidest (Tabel 2) mida genereeritakse uue projekti loomisel kasutades NestJs CLI käsurea tööriista.

Tabel 2. NestJs CLI poolt genereeritud baas failid.

Faili nimetus	**Kirjeldus**
main.ts	Rakenduse sisenemis fail. Siin luuakse NestJs instants ning käivitatakse rakendusega seotud komponendid.
app.controller.ts	Veebiteenuse ressursi kontroller. Siinkohal defineeritakse ressursid mis on avalikud klientseadmetele päringute tegemiseks.
app.sevice.ts	Klass mis hõlmab endas loogikat kuidas ja mida tuleb tagastada kontrolleri pihta tehtud ressursi päringutele.
app.module.ts	Kapseldav moodul, seob kokku ühe funktsionaalse mooduli veebiteenuses mis koosneb muuhulgas mooduli kontrollerist ja mooduli teenusest ehk „service“ klassist.
app.controller.spec.ts	Testimisklass kontrolleri pihta tehtavate automaat-testide läbiviimiseks. Siia faili saab olemasoleva baas testi juurde lisada ühikteste.
app.e2e-spec.ts	Sarnaselt eelneva klassiga on tegemist automaattestide kirjeldamise klassiga. Siinkohal defineeritakse integratsiooni testid erinevate moodulite vahelise töö korrektse toimimise valideerimiseks
jest-e2e.json	Jest testimisraamistiku konfiguratsiooni fail integratsiooni testide konfigureerimiseks.
nest-cli.json	NestJs raamistiku konfiguratsiooni fail, siin on võimalik konfigureerida CLI valikuid mis on kasutatud projekti siseselt.
package.json	Projekti npm konfiguratsioonifail.
tsconfig.json	TypeScript kompilaatori konfiguratsioonifail.

3.1.2 Andmebaasi ühenduvus

Selleks, et kliendi seisu mingit moodi salvestada sessioonide jooksul, tuleb meil kasutada andmebaasi. Andmebaasi otse ühendada klientrakendusega oleks ebaturvaline, kuna siis oleks kliendil samuti võimalus muuta teiste kasutajate andmeid ning pääseda ligi muudele andmetele millele neil ei ole voli.

Andmebaasi ühendus on loodud veebiteenusega otse ning kasutaja saab andmetele ligi kontrollitud viisil vaid läbi veebiteenuse. Selleks, et luua ühendus MySQL andmebaasiga kasutame TypeORM raamistikku. Tegemist on ORM (Object Relational Mapping) tüüpi raamistikuga, mis on loodud konkreetseks kasutamiseks TypeScript või JavaScript jooksutus keskkondades.

TypeORM võimaldab meil teha otseühendus veebiteenusest andmebaasi ning hallata andmebaasi täielikult läbi koodi. Raamistik on tugevalt inspireeritud muude populaarsete raamistike ORM lahendustest nagu näiteks Hibernate, Doctrine või Entity Framework [41]. Selle kaudu loome esialgselt andmebaasi mudeli ning kõik olemid mis mudelis on defineeritud, salvestatakse „*.entity.ts“ faililaiendiga faili projekti sisse. Igas olemi failis on defineeritud selle konkreetse olemi read ning relatsioonid teiste olemitega. Kõiki neid olemeid saab hiljem pärida andmebaasist ning neid hallata.

Siin on oluline teha märkus, et reaalses toimivas rakenduses me ei taha, et defineeritud andmebaasi olemid automaatselt kirjutatakse üle nendega mis on juba olemas andmebaasis. Selleks on võimalus keelata andmebaasi mudeli automaatne uuendamine läbi TypeORM-i. Selleks, et hiljem uusi olemi muudatusi sisse viia, tuleb teha migratsioone. Peale igat olemite muudatusi tuleb genereerida TypeORM käsurea tööriista abil uus migratsioon ning see käivitada, et muudatused andmebaasis aktiveeruksid. Selle abil me ei kaota muudatustest tingitult väärtuslikke kasutaja andmeid või äriloogika poolt genereeritud andmeid, kui mõni olem peaks muutuma uute funktsioonide lisamisel [41].

Näide ühest olemifaili sisust on välja toodud alljärgnevalt Joonis 7 juures, seal on tegemist kasutajate grupi olemiga millel on relatsioonid omakorda teistele olemitele.

```
@Entity()
export class Group extends BaseEntity {

    @ManyToMany(() => User, user => user.groups)
    users: User[];

    @Column({nullable: true, unique: true,})
    name?: string;

    @Column({nullable: true})
    description?: string;

    @Column({nullable: true})
    pictureId?: number;

    @ManyToOne(() => Picture)
    picture?: Picture;
}
```

Joonis 7. Olemi „Group“ sisu koodifailis.

Joonistelt on näha, et kasutatakse annotatsioone nagu „@ManyToOne“ ja „@ManyToMay“ mille abil koostatakse relatsioonid koodis erinevate olemite vahel.

3.1.3 Rest API

Kõik ressursid mille pihta klientrakendus saab päringuid teha on kirjeldatud „*.controller.ts“ laiendiga failides. Kontrollerites märgistatakse funktsioonid vastavalt kas @GET, @POST, @PUT või @DELETE annotatsioonidega. Valikuliselt lisatakse ressursi kontrollerile sõnekujuline aadress, millele klientrakendus peab päringu korral viitama.

Kontrolleri ressursid saavad olla avalikud ja kaitstud. Kui ressurss on avalik, tähendab see seda, et puudub igasugune autentimine kliendi identiteedi kinnitamiseks. Kaitstud ressurss saab olla piiratud väga erinevate kriteeriumite alusel nagu näiteks kasutaja roll, kasutajanimi, aktiivse sessiooni olemasolu ning muu sarnane.

Ressursi päringud tagastavad konkreetse ressursi DTO (Data Transfer Object) mille klient JSON kujul kätte saab. Samuti tagastatakse ka tagastuskood mis automaatselt pannakse päringule kaasa kontrolleri poolt (saab ka manuaalselt defineerida) kas eduka või ka nurjunud ressursipäringu korral. Üldised tagastuskoodid on eduka päringu puhul 200, 201, 203. Ümbersuunamise korral on 300-ga algavad koodid. Tõrgete või tühjade vastuste puhul on 400, 401, 403 või 404. Serveri vea tõttu tagastatakse veakood milleks on 500.

Järgnevalt on näidatud (Tabel 3) peamised veebiteenusesse loodud päringu ressursid. Iga päringu ette tuleb panna omakorda sõne „v1/api“, mis viitab veebiressursi versioonile ning asjaolule ,et tegemist on just veebiteenuse ressursiga.

Tabel 3. Veebiteenuse peamised ressursid.

Ressurss	**Päringu tüüp**	**kirjeldus**
/auth/register	POST	Kasutaja registreerumine
/auth/login	POST	Kasutaja sisse logimine
/get-games-list	GET	Mängude nimekiri sisselogitud kasutajale
/get-games-list-public	GET	Mängude nimekiri anonüümsele kasutajale
/get-game	GET	Konkreetse mängu sisu
/resolve-game	POST	Mängu lahendamise päring
/resolve-game-public	POST	Anonüümne mängu lahendamise päring
/user-data	POST	Kasutaja profiili uuendasime päring
/user-data	GET	Kasutaja profiili informatsiooni päring
/group-data	GET	Kasutaja grupi kohta andmete päring
/add-group	POST	Kasutajate grupi loomise päring
/remove-group	POST	Kasutajate grupi eemaldamise päring
/add-group-user	POST	Alamkasutaja loomise ja gruppi paigutamise päring
/remove-group-user	POST	Alamkasutaja grupist eemaldamise ja kasutaja kustutamise päring

Peale väljatoodud ressursside on veel olemas CRUD (*Create Read Update Delete*) ressursid kõigi olemite jaoks. Neile on peamiselt ligipääs vaid administraatorkasutajal.

3.1.4 Turvalisus

Nest.Js pakub integratsioone populaarsete Node.Js autentimis raamistikega nagu Passport. Kontrollerite ressurss on kaitstud kasutades JWT (JSON[1] Web Token) lahendust, et lubada ligipääs vaid autenditud kasutajatele.

Kõik kasutajate paroolid on turvaliselt hoiustatud andmebaasis soolatud räsi formaadis. Räsi koostamise kasuta kasutades aes-256 šiffrit, millele lisatakse soolamis tsükleid. Räsi koostamise ning paroolide võrdlemise jaoks kasutatakse Node.js sisse ehitatud „crypto“ moodulit.

Omakorda pakub NestJs filtreid, mis aitavad ressurssi tõhusamalt piirata või lubada konkreetsematele kasutajatele või kasutajarollidele läbi NestJs „Guard“ ja „Filter“ moodulite. Alljärgnevalt on näitena välja toodud üks kontrolleri ressursi definitsioon, millel on näidatud rolli autoriseerimise annotatsiooni kasutamist (Joonis 8).

```
@Roles(Role.User, Role.Admin)
@Get('group-data')
getGroupData(@Req() req) {
    return this.gameEndpointsService.getGroupData(req.user);
}
```

Joonis 8. Kontrolleri ressursi rolli põhine autoriseerimine.

3.1.5 Sisend andmete validatsioon

On hea tava valideerida kõiki andmeid, mida klient saadab veebiteenuse pihta, et olla kindel saadetud andmete korrektsuses. Selleks, et kontrollida kas andmed mis saadetakse veebiteenuse ressursi pihta on need mida ressurss ise ootab, pakub NestJs andmete validatsiooni tuge. NestJs on väga paindlik selles osas, on võimalik valideerida kas sisend on kindlat tüüpi kasutades annotatsioone kontrolleritel või andmeobjektides endas. Võimalus on keelata valede andmete sisestamine ning automaatselt tagastada veakood puuduva või vale välja kajastamiseks. Veel on võimalus lülitada sisse andmete automaatne validatsioon, kus vastavalt konfiguratsioonile on näiteks võimalus ülearused sisestatud andmed kärpida päringust, et kontroller võtaks vastu vaid need andmed mis

[1] *JavaScript Object Notation* – kergeloomuline andmetranspordi ja andmevahetus formaat

vastavad sisend andmetüübile [40]. Näide andmete validatsioonist andmeobjektis on välja toodud Joonis 9 juures.

```
import {IsDefined, IsOptional, IsString} from "class-validator";

export class CreateAddressDto {

    @IsOptional()
    @IsString()
    addressLine?: string;

    @IsOptional()
    @IsString()
    city?: string;

    @IsDefined()
    @IsString()
    country: string;

    @IsOptional()
    @IsString()
    postcode?: string;
}
```

Joonis 9. Andmeobjekti valideerimiseks seatud annotatsioonid.

3.1.6 Domeenivahelised päringud

Kuna loodav keskkond koosneb kahest eraldiseisvast rakendusest, kliendist ja veebiteenusest, on keskkonna edukas toimimine seotud õnnestunud rakendustevahelisest kommunikatsioonist. Kuna nii veebiteenus, kui ka klient mõlemad paiknevad erinevatel domeenidel, on nendevaheline kommunikatsioon vaikimisi takistustega. *Cross-origin resource sharing* ehk CORS on mehhanism mis lubab turvatud domeenidevahelist kommunikatsiooni. Selle abil on meil võimalus pärida ressursse domeenilt mis on erinev pärija domeenist.

NestJs pakub sisse ehitatud tuge CORS mehhanismi aktiveerimiseks veebiteenuses ilma erilise konfiguratsioonita. Vaikimisi piisab lihtsa „.enableCors()“ funktsiooni välja kutsumine rakenduse peaklassist, kuid vahel sellest ei ole piisav. Kui on vaja lubada või keelata ressursi jagamist kindlatel domeenidel või konfigureerida muid spetsiifilisi parameetreid päringu päistes, lubab NestJs sisestada konfiguratsiooni viimasesse funktsiooni, näide sellest on välja toodud juures.

```
.enableCors({
    origin: [
        /^(.*)/,
    ],
    methods: 'GET,HEAD,PUT,PATCH,POST,DELETE',
    preflightContinue: true,
    optionsSuccessStatus: 200,
    credentials: true,
    allowedHeaders: [
        'Origin',
        'X-Requested-With',
        'Content-Type',
        'Accept',
        'Authorization',
        'authorization',
        'X-Forwarded-for'
    ],
})
```

Joonis 10. CORS mehhanismi aktiveerimine NestJs raamistikus.

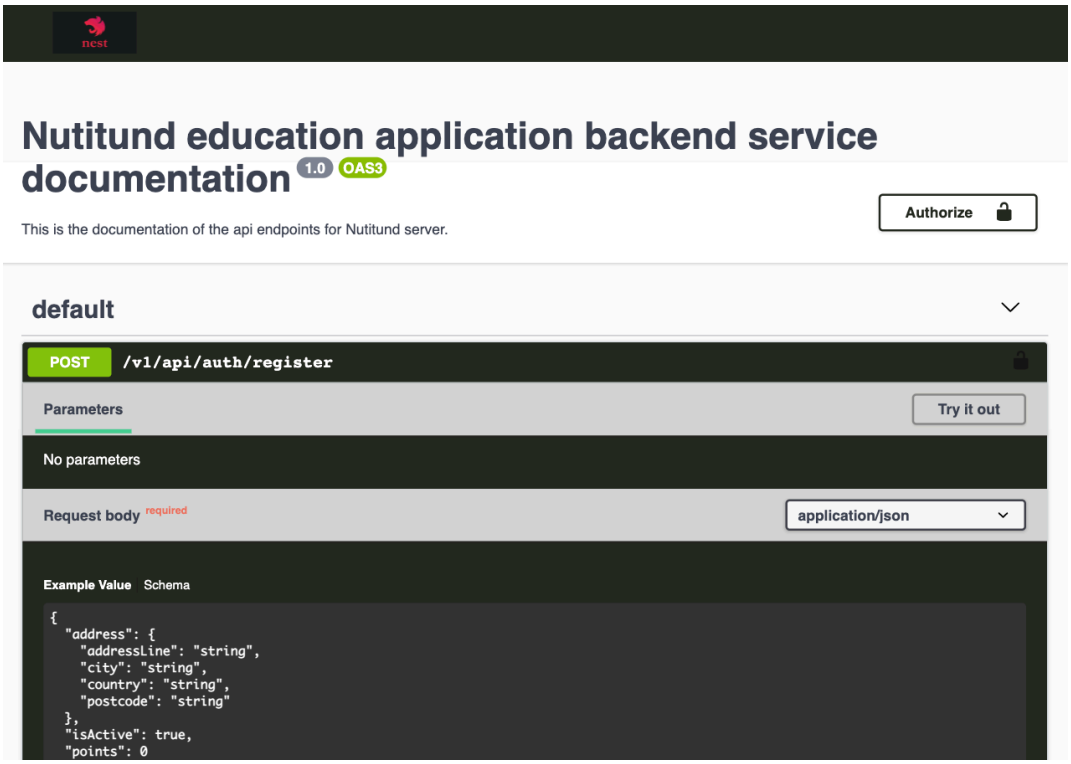
3.1.7 Open API kasutajaliidese tugi (Swagger)

Open API spetsifikatsioon on RESTful veebiteenuste dokumenteerimis ja arendust kiirendav tööriistade komplekt. Open API pakub Swagger kasutajaliidest, mille abil on võimalus genereerida graafiline kasutajaliides kõikide veebiteenuste ressursside jaoks [42]. Nii on mugav näha kõiki olemasolevaid ressursse, mis veebiteenus toetab koos detailse informatsiooniga mis andmeid konkreetne endpoint[1] võtab sisse ning milliseid andmeid see tagastab. Lisaks näeb lisainfot ressursi tagastuskoodide ja autentimisnõuete kohta.

NestJs pakub tuge Swagger kasutajaliidese genereerimiseks, selleks tuleb programmi põhifailis välja kutsuda Swagger loomise funktsioon, millele saab valikuliselt sisestada ka konfiguratsiooni objekti. Lisatingimusena tuleb kõikide andmebaasi olemite juures kasutada TypeORM teegist „BaseEntity“ klassilaiendit, sest just nõnda saab Swagger metaandmeid veebiteenuse ressurssides kasutatavate andmeobjektide kohta.

Näide toimivast Swagger kasutajaliidesest on välja toodud Joonis 11 juures.

[1] Veebiteenuse poolt välja antava ressursi aadress või pääsupunkt



Joonis 11. Swagger kasutajaliides.

3.2 Klientrakendus

Nagu veebiteenuski, on klientrakendus loodud olema iseseisev ning väljavahetatav. Kliendi raamistikuks sai analüüsi käigus valitud Vue.js raamistik. Vue.js pakub madalat õppimiskõverat ning komponendipõhist struktuuri tagamaks puhast programmi struktuuri ning kerget skaleeritavust.

Iga komponent omab iseseisevat funktsionaalsust veebilehel. Komponente saab sisestada üksteise sisse ning neid saab välja vahetada lehtede sees. Ühes komponendis saab olla veel hulk teisi komponente. Iga teine komponent veebilehel saab alguse esimesest „App“ komponendist mis sisestatakse raamistiku poolt igale vaate lehele.

Iga komponent on tähistatud faililaiendiga „*.vue“. Komponent sisaldab endas funktsionaalsust, mis on iseloomulik just konkreetsele komponendile (TypeScript/JavaSript), stiili (CSS/SCSS) ja struktuuri (HTML). Koodinäide komponendi nimetatud omaduste ja sisu demonstreerimiseks, on välja toodud programmikoodi näite Joonis 12 juures.

```
<template>
  <img alt="Vue logo" src="./assets/logo.png">
  <HelloWorld msg="Welcome to Your Vue.js App"/>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
  setup() {
    alert('Hello World!');
  },
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

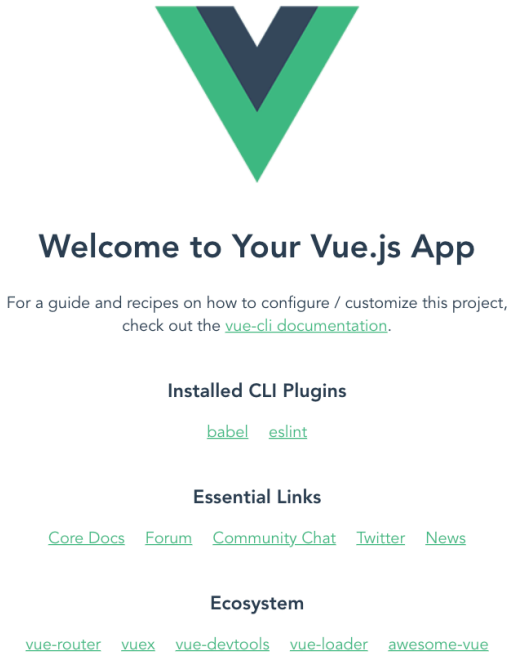
Joonis 12. Vue.Js komponendis sisalduvad koodiblokid: struktuur (template), loogika (script) ja stiil (style).

Näites on näha kuidas väline komponent imporditakse ning seda kasutatakse komponendi struktuuri blokis. Sellist viisi on võimalik paigutada väliseid komponente vastavalt soovile või vajadusele olemasoleva komponendi struktuuri sisse.

3.2.1 Projekti loomine

Autor kasutas Vue.js klientrakenduse loomiseks Vue CLI tööriista. Vue CLI on standardtööriist Vue.js projektipõhja genereerimiseks. Tööriist pakub installeerimis valikuid käsurealt, kui ka graafiliselt liideselt. Genereeritud projekt pannakse automaatselt tööle kohaliku veebiserveriga mille saab avada vaikimisi aadressilt „localhost:8080“.

Projekti genereerimise käigus saab teha mitmeid valikuid konfiguratsiooni osas. On võimalus valida kas projekt arendatakse kasutades TypeScript-i, milline on eelistatud koodiformaatimis reeglistik, milline on vaikimisi paketihaldus raamistik (npm) ning muud sarnast. Genereeritud failide hulka kuuluvad alustuseks näitekomponendid mida kuvatakse välja veebiserveri aadressil. Näide rakenduse avakuvast kohe peale projekti genereerimist, on välja toodud Joonis 13 juures.



Joonis 13. Klientrakenduse avakuva kohe peale uue Vue.Js projekti genereerimist.

Tähelepanu tuleb pöörata asjaolule, et veebiserver koos serveeritava rakendusega toimib esialgselt arendusrežiimis (*development mode*) mis sobib hästi projekti arendamise jooksul kasutamiseks.

Selleks, et hiljem juba valmis rakendus välja serveerida laiemale publikule, tuleb käsurealt käivitada käsk „npm run build“ mis genereerib optimeeritud projekti failid kausta nimega „dist“. Selle kausta sisu ongi lõpp produkt, selles olev „index.html“ fail koos ülejäänud sisuga, tuleb välja serveerida rakenduse majutusserveris olevast veebiserverist [25].

Kuna väljundfailid on staatilised, ehk peamine fail mis avatakse on index.html, ei ole tarvis eraldi veel jooksutuskeskkonda Node.Js raamistiku olemasolu, nagu seda on vaja NestJs veebiteenuse jooksutamise juures. See annab võimaluse loodud andmepaketti vabalt kasutada näiteks CDN tüüpi andmehoidlates, mis teeb võimalused klientrakenduse kättesaadavaks tegemiseks klientidele väga paindlikuks.

Järgnevalt on välja toodud tabel mis sisaldab põhilisi faile (Tabel 4) mida genereeritakse uue projekti loomisel kasutades Vue.Js CLI käsurea tööriista.

Tabel 4. Vue.Js CLI poolt genereeritud baasfailid.

Faili nimetus	**Kirjeldus**
public/index.html	Rakenduse peamine struktuuri fail. Siia sisestatakse Vue poolt kõik rakenduse käivitamisega seotud komponendid või komponentide puu. Antud fail serveeritakse välja arenduse jooksul lokaalse veebiserveri poolt arendatava projekti kuvamiseks.
src/assets/	Kaust kuhu vaikimisi pannakse kõik rakendusega seotud staatilised meedia failid nagu pildifailid või helifailid.
src/components/main.ts	Rakenduse sisenemis fail. Siin luuakse Vue.Js instants ning käivitatakse rakendusega seotud komponendid.
src/components/App.vue	Vue.Js poolt genereeritud näite komponent mis kapseldab endas kõiki kolme põhi komponenti: stiil (CSS), struktuus (HTML) ja loogika (TypeScript).
babel.config.js	Babel kompilaatori konfiguratsiooni fail.
Package.json	Projekti npm konfiguratsiooni fail.
README.md	Projekti dokumentatsioonifail

3.2.2 Lehekülgede kujundus

Lehtede kujundus kasutades CSS kujunduskeelt võib olla aeganõudev ja tekitada palju koodi, mis on arendaja enda nägu. Et hoida ühtset lähenemist komponentide loomisel ja kujundamisel on autor kasutusele võtnud Bootstrap raamistiku. Bootstrap aitab meil kiirendada komponentide ja lehekülgede kujundust hoides samal ajal puhast struktuuri. Bootstrapi sees on defineeritud hulk juba valmis loodud komponente (nupud, vormid, konteinerid) mida saab oma rakendusse kiirelt sisestada ilma, et oleks vaja ühtegi uut klassi luua.

Üks väga oluline funktsioon loodaval kesskonnal on selle võime kohanduda erinevate ekraani suuruste ja resolutsioonidega. Bootstrap pakub lihtsat viisi kohandada veebilehte kõikidele ekraanisuurustele (mobiil, tahvel, töölaud) kasutades nende „container“ komponente koos lisa märgendiga „-sm“, „-md“ või „-lg“ mis aktiveeruvad, kui ekraani kuvasuhe on vastavalt kitsas (mobiil), keskmine (tahvel) või lai (töölaua vaade) [28].

3.2.3 Lehesisene navigatsioon

Vue.js pakub tuge erinevate lehtede vahelisel navigatsioonil läbi nende poolt loodud Vue Router teegi. See ei ole mitte ainus viis kuidas leheküljesisest navigatsiooni läbi viia, kuid üks populaarsemaid ning ametlikult soovitatavatest.

Kasutades Vue Router-it on meil kerge vaevaga võimalus luua lehesisene navigatsioon. Navigeerida on võimalik paigutades „router-link“ elemente HTML struktuuri sisse või manuaalselt koodis kutsudes välja „router.push()“ funktsiooni soovitava lehekülje nimega millele on vaja navigeerida [43].

3.2.4 Komponendi oleku haldus

Komponendi oleku haldus ehk *state management* on oluline lüli Vue.Js komponentide loomisel. Oleku all peetakse silmas asjaolu, et komponendis on muutujad, mille väärtust ja seisundit antud komponent jälgib. Komponendid saavad olla olekuga kui ka olekuta (stateless). Näide olekuga komponendist on registreerumis vorm, kuhu peale igat sisestatud vormivälja salvestab komponent väljade sisse kirjutatud teksti. Kinnitamis nupule vajutades loeb komponent salvestatud oleku vormitekstid ning saadab selle edasi päringuna serverile. Olekuta komponent saab olla näiteks pildigalerii või staatiline tekst koos piltidega, mis ei muutu vastavalt kasutaja sisenditest [25].

Vue.Js rakendusel saab olla peamiselt kaks erinevat olekut: komponendi sisene olek ja globaalne olek. Globaalne olek on jagatud kõikide loodud komponentide vahel ning iga komponent võib sinna kirjutada või sealt lugeda.

Oleku paremaks haldamiseks on Vue välja pakkunud tööriista nimega VueX. Selle eesmärk on pakkuda standard lähenemine komponentide globaalse oleku haldamiseks [44]. Autor nimetatud tööriista projektis palju ei kasuta, kuna leiab, et selle kasutus loob liigseid samme kergeloomuliste andmete salvestamiseks globaalsesse olekusse. Sama

funktsionaalsus on võimalik saavutada eksportides üldine oleku klass globaalseks kasutamiseks kõigile projekti komponentidele.

3.2.5 Kommunikatsioon veebiteenusega

Autor on veebiteenusega kommunikeerimiseks kasutusele võtnud Axios teegi. Axios pakub HTTP klienti mille abil saab veebiteenuse pihta teha GET, PUT, POST, DELETE päringuid koos päringu andmetega. Axios pakub laialdasi võimalusi erinevate parameetrite muutmiseks päringute tegemisel. Päringutele saab vaikimisi kaasa anda „HTTP Header“ konfiguratsioonid ning muid turvalisusega seotud andmeid. Axios toimib koos „promise“ funktsioonidega, mis muudab mugavaks päringute vastuste ootamise ilma, et oleks vajadust tagastada „call-back“ tagastusfunktsioone koodi loetavuse ja puhtuse tõstmiseks [45].

3.2.6 Turvalisus

Klient suhtleb veebiteenusega kasutades HTTP või HTTPS (*HyperText Transport Protocol over SSL*) protokolli. Kui kasutaja logib sisse klientrakenduses, saadetakse tema kasutajaga seotud informatsioon läbi interneti veebiteenuse pihta. Taoline liiklus võib panna kasutaja ja ka veebiteenuse ohtu, kui võrgul on pealtkuulajad.

Turvalise autentimise võimaluste saavutamiseks on kasutusele võetud JWT. Kui kasutaja logib sisse siis tagastatakse temale veebiteenuse poolt JWT kood, mille ta peab saatma iga järgneva päringuga kaasa veebiteenusele. Nii peab veebiteenus järge kasutaja sessiooni aegumise ja õiguste kohta ning vastavalt annab kasutajale märku kui tema sessioon on aegunud või katkenud. Vastavalt peab siis kasutaja uuesti sisse logima, et saada uus JWT kood mida salvestada lokaalselt. JWT salvestamiseks lokaalselt klientrakenduses kasutatakse veebilehitseja lokaalset andmehoidlat ehk *local storage* funktsionaalsust.

3.2.7 Piltide hoiustamine

Kuna Nutitund keskkonnas esineb mängude ja ülesannete juures palju pilte, on mõistlik planeerida, kuhu meediafaile salvestada ning kuidas neid kätte saada kasutajale kuvamiseks.

Üheks viisiks on salvestada kõik pildid veebiteenusesse ning anda kaasa Base64 andmetena igale päringule, mis vajab pildifaili. See aga oleks võrguressursi ja veebiteenuse ressursi ebaoptimaalne kasutamine sellise andmesisu jagamiseks.

Teiseks on võimalus talletada pildifailid andmebaasi ning seal sarnasel viisil nagu eelmine tagastada need kliendile päringus. Ka see lähenemine utiliseeriks palju võrguliiklust meediafailide jagamiseks kliendile muutes ka andmebaasi koormust päringute tegemisel suuremaks.

Kolmandaks oleks võimalus pildi- ja helifailid salvestada otse klientrakenduse sisse, ning neid kasutada sealt otse vastavalt vajadusele, sellise lähenemise puhul toome pildifailide haldamiskoormuse andmebaasi ja veebiteenuse pealt klientseadme ja klientrakenduse peale. Selline lähenemine oleks rohkem soovitav, sest nii jääb rohkem ressurssi üle veebiteenusel rohkemate klientide teenindamiseks.

Viimase lähenemise puhul oleks andmebaasis kirjeldatud millist pilti näidata konkreetse mängu juures, ehk on kirjas pildi ID või selle nimi. Kui klientrakendus saab veebiteenuselt päringuna vastuse mängude konfiguratsiooni, võtab klient salvestatud piltide seast vastava nimega pildifaili ning kuvab seda antud mängu sees.

Lisaks on ka võimalus salvestada pildifailid kolmandate osapoolte CDN lahendustesse, kuid kulude madalal hoidmise huvides jätab autor nimetatu väljapoole valikutest.

3.3 Testimine

Nii klientrakendus kui ka veebiteenuse juures on raamistikes endas olemas automaattestimis tööriistade tugi. Mõlema puhul on vaikimisi toetatud Jest automaattestimis raamistik loodud komponentide funktsionaalsuse testimiseks. Jest-i abil on ka võimalus teha integratsiooni teste, valideerides keskkonna osade õiget toimimist erinevate tingimuste juures.

Projektidesse on kirjutatud algne testide kogu, kindlasti on see ala, mida saab ning tuleks edasi arendada projektikoodi kvaliteedi tõstmiseks ja üleval hoidmiseks. Peamised testid loodud keskkonna juures on seni tehtud manuaaltestimis vormis, see tähendab käsitsi läbi käies ning läbi kontrollides loodud funktsioonide korras olekut ja eesmärgikohast toimimist.

Peale automaattestide laiendamise on ka rõhk edasi arendada automaat UI (*User Interface*) teste kasutades näiteks sarnast raamistikku nagu Cypress või Nightwatch.

3.4 Pidev integratsioon

Selleks, et automatiseerida keskkonna ehitamist, testimist ja serveerimist lõppkasutajale, on autor kasutusele võtnud GitLab-i tegumijooksutajad. Nende kasutamisel saab peale koodimuudatuste salvestamist koodirepositooriumisse käivitada automaatselt ehitamise protsess. Protsessi jooksul on võimalus käivitada rakenduse automaattestid ja integratsioonitestid valideerimaks rakenduse toimimist ning terviklikkust. Peale testide edukat läbimist käivitatakse rakenduse paketi ehitamise protsess.

Kui pakett on loodud luuakse rakendusest Docker *image*, mis omakorda salvestatakse automaatselt koodirepositooriumis olevasse konteinerite registrisse. Järgmise sammuna tehakse automatiseeritud uuendus majutatud rakendusele CapRover keskkonnas, uuendades rakenduse *image* versiooni sellega, mis sai loodud eelmise sammu juures. Järgnevalt aktiveerub CapRover-is uue Docker konteineri loomise ja käivituse protsess, kus sisalduvad uued koodimuudatused mis said salvestatud vastava Docker image sisse.

Kuna GitLab on vabavaraline lahendus, ei küsi nad tasu nende teenuste kasutamise eest, siiski tuleb meeles pidada, et nende poolt jagatud erineval ressursil võib olla rakendatud ajaline kasutuslimiit, näiteks kuu kasutuslimiit. Jagatud tegumijooksutajate (pipelines) kasutuslimiidi täitumisel ei ole võimalik enam projekti ehitamist ja serveerimist automatiseerida. Selleks annab GitLab võimaluse kasutada oma enda privaatseid GitLab *runner* instantsi, mille abil kasutatakse oma serveri ressurssi tegumijooksutajate jooksutamisel ning täiendavaid limiite konkreetse ressursi koha ei rakendu. Kuna püsivat koormust see serverile ei avalda, vaid hetkeline ehitus protsessi ajal, on otsustanud autor käivitada lisakonteineri CapRover keskkonnas, mis tegeleks just GitLab keskkonnas käivitatud automatiseerimisprotsesside jooksutamisega.

Selleks, et käivitada kogu automatiseerimisprotsess tuleb luua projekti juurkausta fail mille nimeks „gitlab-ci.yml“. Faili sisse peab kirjutama käsud, mis vastavalt aktiveeriks automaattestid, rakenduse ehitus ja majutusprotsessid. Näide development keskkonna automatiseerimis scriptist on välja toodud alljärgneva Joonis 14 koodinäite juures.

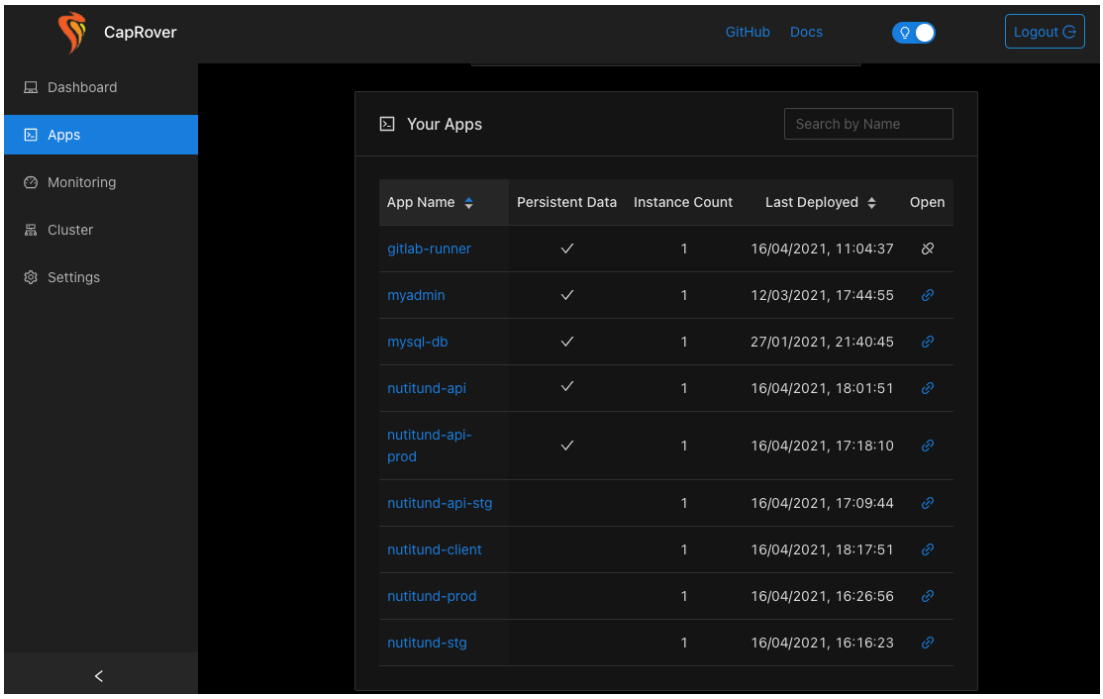
```
build-docker-master: # dev environment
  image: docker:19.03.1
  stage: build
  services:
    - docker:19.03.1-dind
  before_script:
    - export DOCKER_REGISTRY_USER=$CI_REGISTRY_USER # built-in GitLab Registry User
    - export DOCKER_REGISTRY_PASSWORD=$CI_REGISTRY_PASSWORD # built-in GitLab
Registry Password
    - export DOCKER_REGISTRY_URL=$CI_REGISTRY # built-in GitLab Registry URL
    - export COMMIT_HASH=$CI_COMMIT_SHA # Your current commit sha
    - export IMAGE_NAME_WITH_REGISTRY_PREFIX=$CI_REGISTRY_IMAGE # Your repository
prefixed with GitLab Registry URL
    - docker login -u "$DOCKER_REGISTRY_USER" -p "$DOCKER_REGISTRY_PASSWORD"
$DOCKER_REGISTRY_URL # Instructs GitLab to login to its registry

  script:
    - export CONTAINER_FULL_IMAGE_NAME_WITH_TAG=$IMAGE_NAME_WITH_REGISTRY_PREFIX:dev
    - docker build -f ./Dockerfile --pull -t built-image-name .
    - docker tag built-image-name "$CONTAINER_FULL_IMAGE_NAME_WITH_TAG"
    - docker push "$CONTAINER_FULL_IMAGE_NAME_WITH_TAG"
    - echo $CONTAINER_FULL_IMAGE_NAME_WITH_TAG
    - docker run caprover/cli-caprover:v2.1.1 caprover deploy --caproverUrl
$CAPROVER_URL --caproverPassword
    $CAPROVER_PASSWORD --caproverApp $CAPROVER_APP_NUTITUND_API_DEV --imageName
$CONTAINER_FULL_IMAGE_NAME_WITH_TAG
  only:
    - master
```

Joonis 14. *Development* keskkonna pideva integratsiooni script.

3.5 Keskkonna majutamine

Nii klientrakendus kui veebiteenus ja andmebaas on autori poolt majutatud tellitud VPS keskkonnas toimiva CapRover Paas süsteemi peal. Kõik peale andmebaasi on toimivad Docker konteinerites Paas süsteemis oleva Docker Swarm konteinerhaldus raamistiku peal. Konteinerid luuakse automaatselt kasutades GitLab CI/CD võimalusi, kuid neid saab ka manuaalselt luua ning keskkonda paigutada.

Oluline on ka siinkohal meeles pidada, et nii klientrakendust kui ka veebiteenust serveeritakse kolmekordselt erinevate keskkondade jaoks milleks on development-, staging- ja production keskkond. Andmebaasi puhul on tegemist sama instantsiga, mille sisse on loodud kolm uut andmebaasi erinevate keskkondade kasutamise jaoks. Alljärgnevalt on kuvatud Joonis 15, kus on välja toodud majutatud rakenduste vaade CapRover keskkonnas.



Joonis 15. CapRover keskkonna majutatud rakenduste vaade.

Jooniselt on näha nimekirjas olevat „gitlab-runner“ välja. Sealsega on tegemist privaatse GitLab runner konteineri jooksutamisega, et muuta CI/CD tegumijooksutajad võimekamaks GitLab versioonihaldus keskkonnas. Kui kasutada GitLab-i tasuta tegumijooksutajaid võib nende liigse kasutamise korral ette tulla piiranguid kuu ajalimiidi täitumisel. Privaatse tegumijooksutaja puhul piiranguid ei teki ning nende kasutamisel on CI/CD tööde läbiviimine kiirem VPS serveri ressursi jagamise abil.

Uue rakenduse majutamise protsess CapRover keskkonnas on lihtne nende poolt pakutava kasutajaliidese abiga. Sisuliselt tuleb nimetada millise pordi peal majutatav rakendus välja jagatakse veebiserverilt ning kas on soov sellele lisada HTTPS tugi. Peale seda tuleb defineerida mitu aktiivset konteinerit peaks platvorm käivitama konkreetse rakenduse jaoks, nende arvu saab ka manuaalselt hiljem suurendada või vähendada vastavalt soovile või ressursi vajadusele. Viimaks tuleb sisestada kasutatav konteineri *image* nimi, mis tõmmatakse antud projekti puhul alla GitLab konteiner registrist, kuhu on talletatud kõik loodud Docker konteiner *image* failid, mis on seotud Nutitund keskkonnaga. Järgnevalt laetakse konteiner automaatselt alla CapRover-i poolt, käivitatakse see ja serveeritakse välja rakendusele antud nime ja pordi pealt, mis konfiguratsiooni astme juures sai sellele omistatud.

4 Tulemused ja järeldused

Arendustöö lõpuks valmis lastele suunatud arendavate mängude keskkond, mis tehti avalikuks „www.nutitund.ee“ aadressil. Kuna kasutatav Paas keskkond pakub soovi korral https implementeerimist majutatavatele rakendustele ühe nupuvajutusega, oli ka TLS (*Transport Layer Security*) toe lisamine keskkonna rakenduste jaoks lihtne. Klientrakenduse ehitamine Vue.Js raamistikku kasutades ei olnud alati autori jaoks lihtne, reklaamitud madal õppimiskõver võib olla suhteline, kuna konkreetsuste implementeerimine keskkonda võttis vahel kordades rohkem aega, kui selleks oli soovi kulutada. Väga meeldiv üllatus oli autori jaoks Nest.Js raamistikuga veebiteenuse ehitamine, autor on varasemalt kokku puutunud .NET, JAVA Spring ja Express.Js raamistikega veebiteenuse arenduseks ning ei olnud veel varem kohanud nii kergesti kasutatavat ja arusaadavat raamistikku kui nimetatu. Raamistik tundub väga võimekas olles lihtsasti aru saadav ning järgides parimaid praktikaid, mida võib leida koodi arhitektuuri, struktuuri, organiseerituse ja skaleeritavuse praktikatest [46].

Projekti arenduse jooksul said valmis kõik mittefunktsionaalsed ning pea kõik funktsionaalse planeeritud projektiosad. Kuvatõmmiseid loodud keskkonnast on võimalus vaadelda Lisa 3. juures.

Siiski on mõningaid skoobis olnud komponente mida ei õnnestunud antud aja jooksul valmis saada, suurem nende seast kasutajatevaheline punktide peale võistlemine või kasutajatevaheline koostöö saada ülesanded lahendatud ennem kui ajalimiit kätte jõuab. Viimase mängureziimi eesmärgiks on õpetada lastele meeskonnatööd. Veel jäi keskkonda implementeerimata autentimine kasutades sotsiaalvõrgustike tuge. Nimetatuid funktsionaalsusi saab projekti edasiarenduse juures lõpule viia ning laiendada ka olemasolevaid mänge, et toetada näiteks uusi aineid nagu keemia või füüsika suurtematele lastele lahendamiseks.

4.1 Lastele testimiseks andmine

Loodud keskkond anti testimiseks lastele kes käivad erinevates Eesti lasteaedades. Kuna keskkond on kõigile avalikult kätte saadav aadressilt „www.nutitund.ee“, tuli vaid informeerida lapsevanemaid ja õpetajaid selle olemasolust ning saada tagasiside kasutuskogemusest.

Vastanute seast tuli esile üldine arvamus, et keskkonna olemasolust leitakse kasu laste arengu toetamiseks. Leiti, et mänge ja ülesandeid võiks olla rohkem ning et laste tähelepanu hoidmiseks oleks vaja midagi enamat, kui seda pakub punktisüsteem või olemasolev visuaal. Tagasisides kinnitati, et laps saab hakkama keskkonnas iseseisvalt navigeerimisega, või vajadusel tuleb vähesel määral abistada täiskasvanu poolt. Kinnitati ka, et laps saab temale esitatud küsimusest või probleemist aru ning oskab kasutada sealset kasutajaliidest vastamiseks. Kokkuvõtlikult oldi loodud keskkonna sisu ja omadustega rahul ning soovitakse keskkonna laiendamist, et lastel oleks põnevam keskkonda kasutada.

4.2 Arendusel kasutatud tehnoloogiate uudsus

Keskkonna arendamiseks sai valitud tehnoloogiaid mis kas olid kõige populaarsemad antud töö kirjutamise ajal või mis on kogunud järsku populaarsust viimaste aastate jooksul. NestJs ei ole kõige populaarsem veebiteenuse jaoks kasutatav raamistik, mis hetkel on kättesaadav, kuid omab järsku populaarsuse tõusu viimaste aastate jooksul. NestJs kasutab sisemiselt Express.Js raamistikku, mis on üks populaarseimaid veebiteenuse raamistikke [22]. NestJs annab Express.Js-ile rohkem struktuuri ja häid arenduspraktikaid mida leidub teistest populaarsetest raamistikest, et rakenduse arendus oleks rohkem organiseeritud ja projekti suurenemisel oleks selle haldamine sama kerge kui alguses [40].

Klientrakenduse arendusel sai valitud Vue.Js raamistiku viimane versioon ehk „3.x“. Raamistik on hetkel kõige populaarsem arendajate seas lähtuvalt Github repositooriumis pandud tähekeste arvust. Nende uus versioon „3.x“ toob raamistikku uusi värskendusi, mis pikemas perspektiivis püüavad projektide arendus kiirust ja arendust üldiselt teha veelgi paremaks [25].

MySQL andmebaas on kõige populaarsem andmebaasimootor hetkel kasutuses arendajate poolt ning MariaDB, mis on selle vabavaraline asendaja, on kasutuses antud projektis. Andmebaas ise on kergeloomuline ning funktsioonirikas, võrreldes PostgreSQL andmebaasiga, mis on samuti väga võimekas kuid raskemaloomulisem [30]. Käesolevas projektis MariaDB kasutamine on rohkem õigustatud arvestades selle vajadust olla kergeloomuline, et hästi toimida piiratud serveriressursi ja eelarve juures.

Keskkonna arendusel on valitud raamistike juures kasutatud vaid standard tööriistu või tööriistu mis on leidnud suurt populaarsust ning ühilduvad hästi konkreetse raamistikuga. Nende seast võiks näitena välja tuua Bootstrap integratsioon Vue.Js raamistikus või Swagger kasutajaliides veebiteenuse dokumentatsiooni kuvamiseks veebiteenuse juures.

Populaarsuse hindamisel on kasutatud vastava tehnoloogia Github repositooriumi tähekeste arvu ning analüüsides viimase aasta Stackoverflow arendajate küsitluse tulemusi [22].

4.3 Edasiarendus võimalused

Edasiselt tuleks esialgses skoobis olnud funktsionaalsused lõpuni implementeerida keskkonda, see aitaks kaasa laste tähelepanu hoidmisele ja tõsta soovi lahendada ülesandeid, kui neid on võimalik teha koos oma rühmakaaslastega. Oluline on ka arendada lõpuni reklaampinna väljaüürimine, et keskkonnal oleks mingisugunegi potentsiaalne sissetulek ennast rahastada ning aidata ülalhoiukulusid katta.

Pikemas perspektiivis on võimalus agregeerida vabavaralisi või avatud kasutusõigusega mänge Nutitund keskkonda, et arendavate ülesannete kõrval oleks ka mõni vaid meelelahutuslik mäng, nii ei oleks lapsel alati kohustus ülesandeid nuputada, vaid võimalus ka vahelduseks veeta keskkonnas aega meelelahutuslikult.

Kuna projekti skoobis oli originaalselt luua vaid eesti keele toega keskkond, siis edasiselt on võimalus tuua sisse i18n ehk mitme keele tugi. i18n[1] all peetakse silmas lokaliseerimist ehk tõlkimist, viies kasutajaliidese kasutaja emakeelde.

Tänu asjaolule, et klientrakendus ja veebiteenus on omavahel eraldatud, toob see võimaluse tulevikus luua uusi klient liidestusi keskkonna jaoks. Võimalus on luua Android või Apple rakenduste poodi vastav äpp keskkonna külastamise jaoks või muudele nutiseadmetele ja platvormidele, mis leiavad laialdast kasutust laste seas.

[1] i18n – internationalization, arv 18 viitab tähtede arvule algus- ja lõputähe vahel

5 Kokkuvõte

Lõputöö eesmärgiks oli koostada keskkond mis toetaks laste individuaalset arengut läbi mängude. Mängud on sisuliselt punktide peale ülesanded, mis aitavad lastel paremini omandada kirjaoskust ja arvutamist. Kuna projekti õnnestumiseks on oluline hoida kasutaja tähelepanu, pidi selle jaoks looma keskkonnale lisaks punktisüsteemi, mille abil mängija saab avada uusi kategooriaid.

Loodud lahendus adresseerib lasteaiaõpetajate poolt välja toodud probleemi, milleks on vähene või puudulik arendavate mängude kättesaadavus eesti keele ruumis. Nüüdseks on lasteaedadel võimalus kasutada olemasolevaid tahvelarvuteid ja muid nutiseadmeid konkreetselt arendava sisu ülekandmiseks lastele. Sisu mis mänguliselt arendaks põhi aluseid, mille sisse kuulub kirjaoskus, lugemisoskus ja arvutamine. Samuti on lastel võimalus individuaalselt jätkata keskkonna kasutamist kodus isiklikel nutiseadmetel, et asendada muud üldist meelelahutuslikku ajaveetmise sisu.

Projekti koostamise käik ning projekti olulised osad on välja toodud lõputöö analüüsis. Samuti on autor kirjeldanud valitud tehnoloogilisi suundasid ja põhjuseid nende rakendamiseks koostatud veebikeskkonnas. Töö lõpuks valmis veebikeskkond avaliku aadressiga „www.nutitund.ee“, kus on kõigil võimalus registreeruda ning lahendada ülesandeid ilma, et oleks vajadust tasuda lisa summasid keskkonna kasutamise eest. Keskkonna kasutajal on võimalus luua kasutajagruppe kuhu luua lastele kasutajakontod, nii on laste andmed kaitstud ning laste ligipääs keskkonda kontrollitud.

Kuigi kõik esialgselt planeeritud funktsionaalsused ei valminud töö tähtajaks mahukuse tõttu, on sellegipoolest põhifunktsionaalsus saavutatud ning projekt võimalik kuulutada õnnestunuks. Alus edasiarenduseks on samuti seatud, pideva integratsiooni ahela kaudu on võimalik automatiseeritult käivitada teste ning loodud projekti pakett installeerida serverile. Igal uuel projektiga liitunud arendajal on võimalus projekti edasisele arengule anda oma panus.

Kasutatud kirjandus

[1] Asana team, „Asana, work on big ideas, without the busywork.," Asana, Inc., 2021. [Võrgumaterjal]. Available: https://asana.com/.

[2] A. Roznovsky, „Choosing a Technology Stack for Web Application Development," 2021. [Võrgumaterjal]. Available: https://light-it.net/blog/choosing-a-technology-stack-for-web-application-development/.

[3] D. Barker, Web Content Management: Systems, Features, and Best Practices, O'Reilly Media, Inc, 2016.

[4] A. Vasylenko, „Microservices vs Monolith: which architecture is the best choice for your business?," N-IX, 2018. [Võrgumaterjal]. Available: www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/.

[5] R. T. Fielding, „Architectural Styles and the Design of Network-based Software Architectures," 2000. [Võrgumaterjal]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[6] F. Halili ja E. Ramadani, „Web Services: A Comparison of Soap and Rest Services," Canadian Center of Science and Education, 2018.

[7] S. Buna, Learning GraphQL and Relay, 2016.

[8] K. Ismail, „JAMstack vs. LAMP Stack vs. MEAN vs .NET: Tech Stacks Compared," 2018.

[9] R. Vaghani, „Introduction to Spring Framework," 2019. [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/introduction-to-spring-framework/.

[10] S. Goswami, „Django vs Ruby on Rails – The Best Choice for Mobile App Development," 2021.

[11] U. Pisuwala, „What makes Laravel the most popular PHP framework?," 2017. [Võrgumaterjal]. Available: https://www.peerbits.com/blog/laravel-most-popular-php-framework.html.

[12] Microsoft, „A tour of the C# language," 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

[13] Python Software Foundation, „About Python," 2021. [Võrgumaterjal]. Available: https://www.python.org/about/.

[14] K. Chandrakant, „Why Choose Spring as Your Java Framework?," 2019. [Võrgumaterjal]. Available: https://www.baeldung.com/spring-why-to-choose.

[15] BetterStack, „How long does it take to learn Node.js?," 2019. [Võrgumaterjal]. Available: https://betterstack.dev/blog/how-long-does-it-take-to-learn-nodejs/.

[16] Ruby community, „About Ruby," 2021. [Võrgumaterjal]. Available: https://www.ruby-lang.org/en/about/.

[17] The PHP Group, „What is PHP?," 2021. [Võrgumaterjal]. Available: https://www.php.net/manual/en/intro-whatis.php.

[18] C. Vega, „Client-side vs. server-side rendering: why it’s not all black and white,“ 2017. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d/.

[19] OpenJS Foundation, „Overview of Blocking vs Non-Blocking,“ 2021. [Võrgumaterjal]. Available: https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/.

[20] X. Mao, „Comparison Between Symfony, ASP.NET MVC and Node.JS Express for Web Development,“ Fargo, 2018.

[21] A. D. Pham, „Developing Back-end of a Web Application With NestJs Framework,“ 2020. [Võrgumaterjal]. Available: https://www.theseus.fi/bitstream/handle/10024/353200/Pham_Duc.pdf?sequence=2&isAllowed=y.

[22] Stackoverflow, „2020 developer survey,“ 2021. [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2020.

[23] M. Kaluža, K. Trosko ja B. Vukelić, „Comparison of Front-end Frameworks for Web Applications Development,“ 2018.

[24] H. Dhaduk, „Best Frontend Frameworks of 2021 for Web Development,“ 2021. [Võrgumaterjal]. Available: https://www.simform.com/best-frontend-frameworks/.

[25] E. You, „Vue.Js Guide,“ 2021. [Võrgumaterjal]. Available: https://v3.vuejs.org/guide/introduction.html.

[26] Semantic UI team, „Semantic UI, User Interface is the language of the web,“ 2021. [Võrgumaterjal]. Available: https://semantic-ui.com/.

[27] I. ZURB, „Foundation, the most advanced responsive front-end framework in the world.,“ 2021. [Võrgumaterjal]. Available: https://get.foundation/.

[28] Bootstrap team, „Bootstrap,“ 2021. [Võrgumaterjal]. Available: https://getbootstrap.com/.

[29] C. Wodehouse, „QL vs. NoSQL Databases: What’s the Difference?,“ 2019. [Võrgumaterjal]. Available: https://www.upwork.com/resources/sql-vs-nosql-databases-whats-the-difference?vt_cmp=11384804789&vt_adg=113089129402&vt_src=google&vt_kw=&vt_device=c&utm_source=google&utm_campaign=11384804789&utm_medium=paidsearch&gclid=Cj0KCQiA7YyCBhD_ARIsALkj54rCXxLTA3O.

[30] S. Dhruv, „Pros and Cons of using PostgreSQL for Application Development,“ 2019. [Võrgumaterjal]. Available: https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/.

[31] J. Mack, „Five Advantages & Disadvantages Of MySQL,“ 2014. [Võrgumaterjal]. Available: https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/.

[32] MariaDB team, „MariaDB Platform, Enterprise Open Source Database,“ 2021. [Võrgumaterjal]. Available: https://mariadb.com/products/mariadb-platform/.

[33] H. Singh, „Oracle Database Advantages, Disadvantages and Features,“ 2020. [Võrgumaterjal]. Available: https://theninehertz.com/blog/advantages-of-using-oracle-database.

[34] Z. Younes, „Gitlab VS Github VS BitBucket. Which one deserve your time ?,“ 2018. [Võrgumaterjal]. Available: https://dev.to/yafkari/gitlab-vs-github-vs-bitbucket-which-one-deserve-your-time-2npm.

[35] GitLab, „Run your CI/CD jobs in Docker containers,“ 2021. [Võrgumaterjal]. Available: https://docs.gitlab.com/ee/ci/docker/using_docker_images.html.

[36] CenturyLink, „Traditional vs PaaS hosting,“ 2021. [Võrgumaterjal]. Available: https://www.ctl.io/developers/blog/post/traditional-vs-paas-hosting.

[37] Avi, „8 PaaS to Build and Host Your Modern Applications,“ 2021. [Võrgumaterjal]. Available: https://geekflare.com/paas-for-modern-apps/.

[38] M. Skog, „Heroku vs self-hosted PaaS,“ 2019. [Võrgumaterjal]. Available: https://www.mskog.com/posts/heroku-vs-self-hosted-paas/.

[39] E. Goncharova, „Monetization strategies in free-tp-play mobile games,“ Lahti University of Applied Sciences LTD, Lahti, 2017.

[40] K. Mysliwiec, „NestJs Documentation,“ 2021. [Võrgumaterjal]. Available: https://docs.nestjs.com/.

[41] TypeORM team, „TypeORM,“ 2021. [Võrgumaterjal]. Available: https://typeorm.io/.

[42] SmartBear Software, „Swagger, API Development for Everyone,“ 2021. [Võrgumaterjal]. Available: https://swagger.io/.

[43] E. You, „Vue Router,“ 2021. [Võrgumaterjal]. Available: https://router.vuejs.org/.

[44] E. You, „What is VueX?,“ 2021. [Võrgumaterjal]. Available: https://vuex.vuejs.org/.

[45] M. Zabriskie, „Axios, promise based HTTP client for the browser and node.js,“ 2021. [Võrgumaterjal]. Available: https://axios-http.com/.

[46] E. Gamma, R. Helm, R. Johnson ja J. Vilissides, Design Patterns : Elements of Reusable Object-Oriented Software, 1997.

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

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

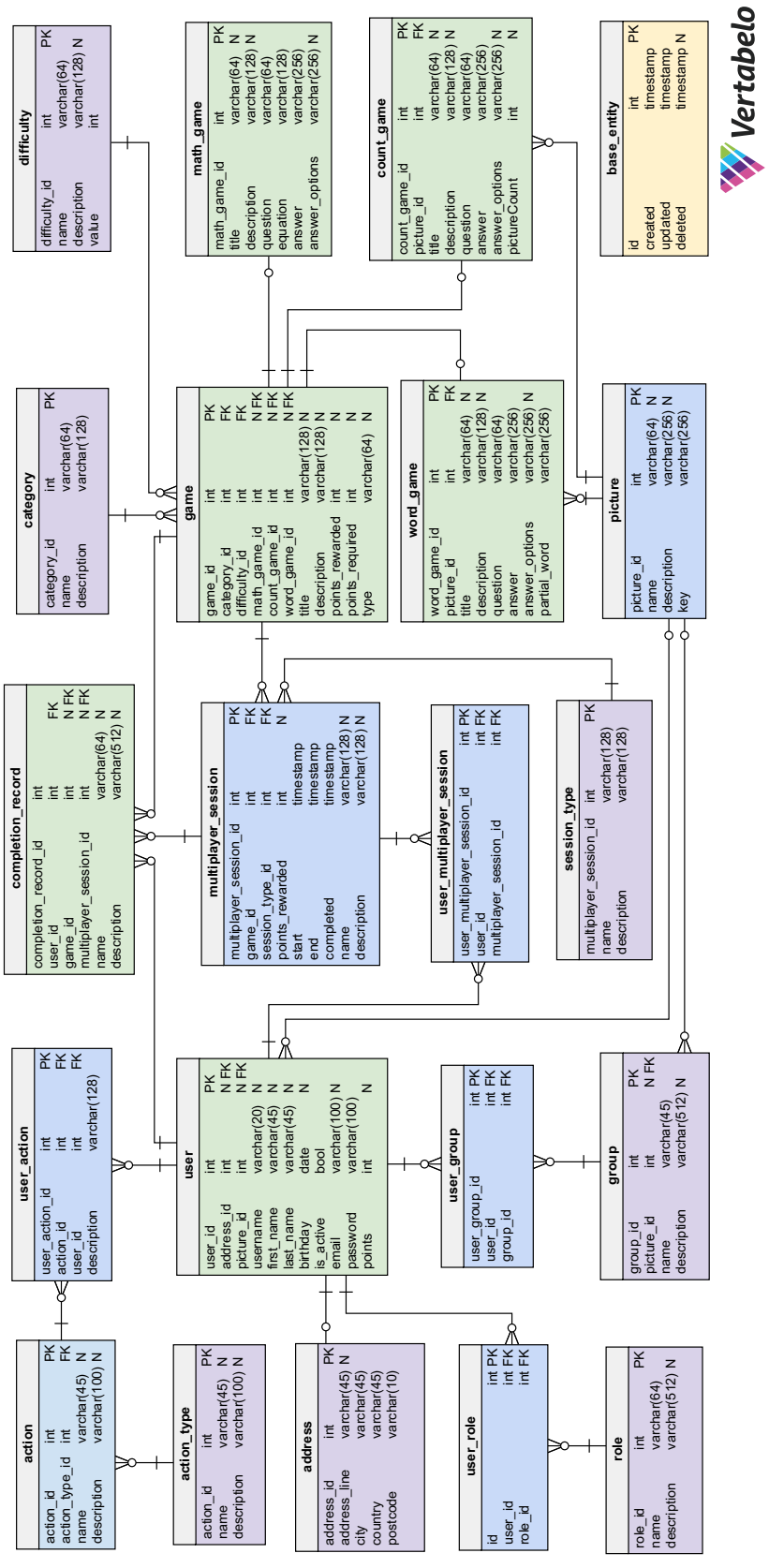
Mina, Taaniel Sülla

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "LASTE ARENGUT TOETAVATE MÄNGUDE KESKKONNA "NUTITUND" ARENDUS", 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.

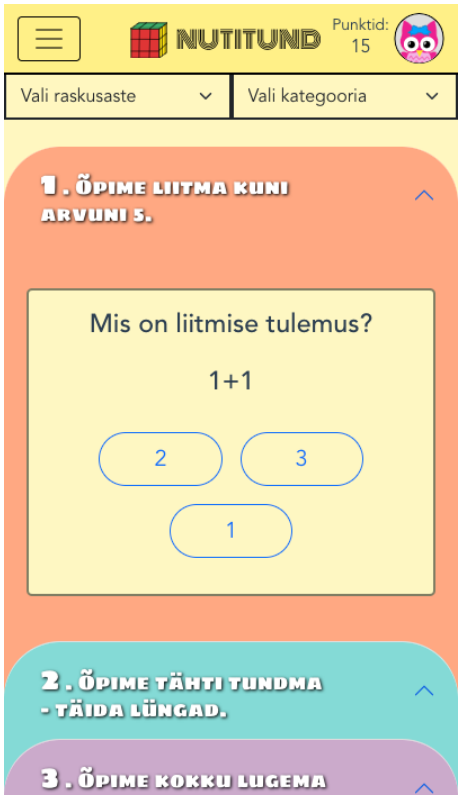
23.04.2021

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.

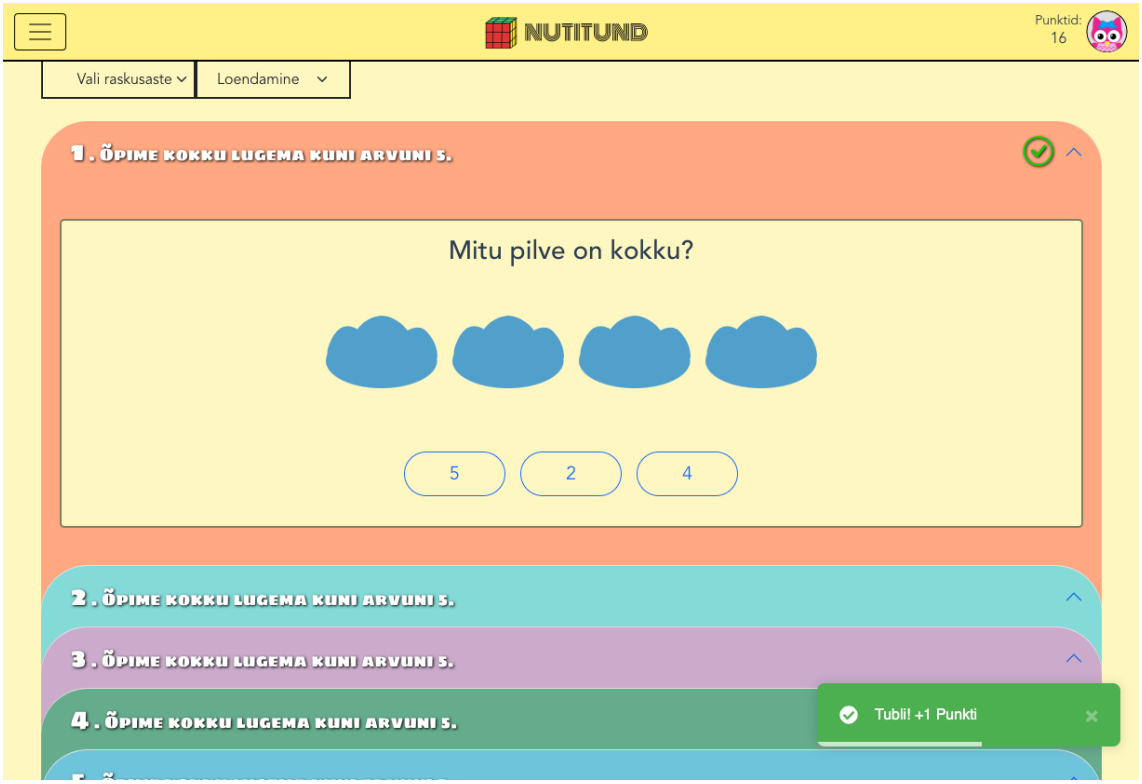
Lisa 2 – Olemi-suhte diagramm

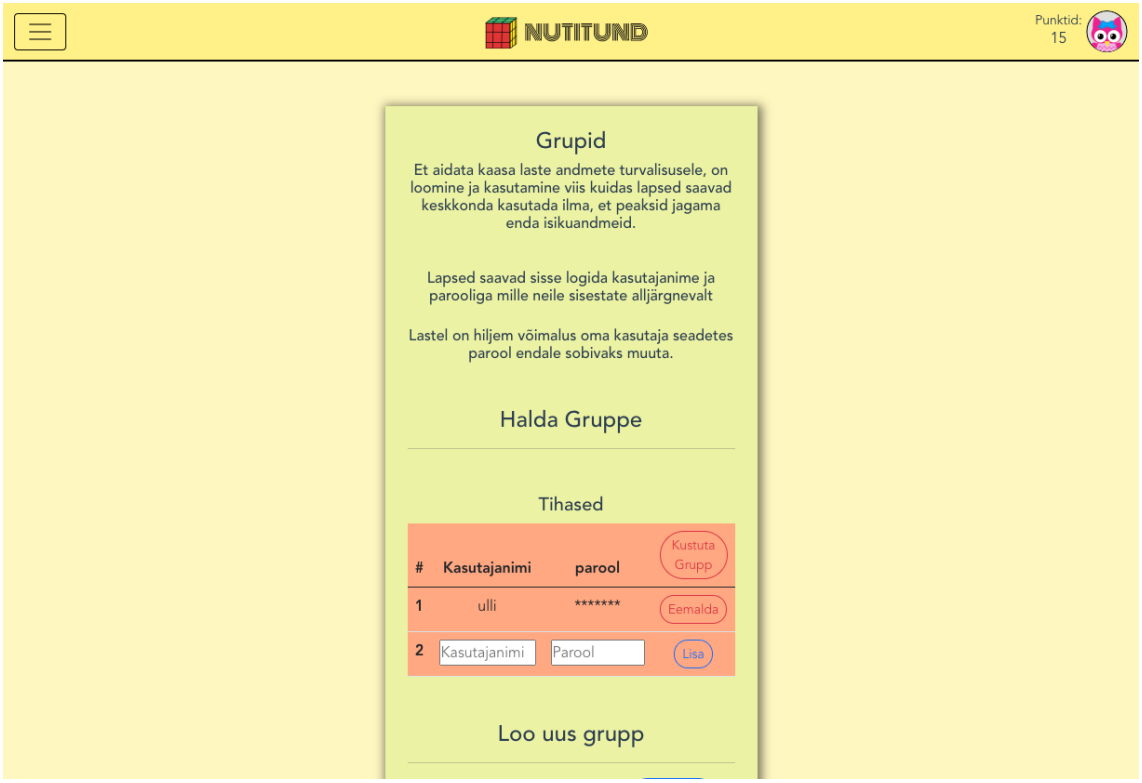


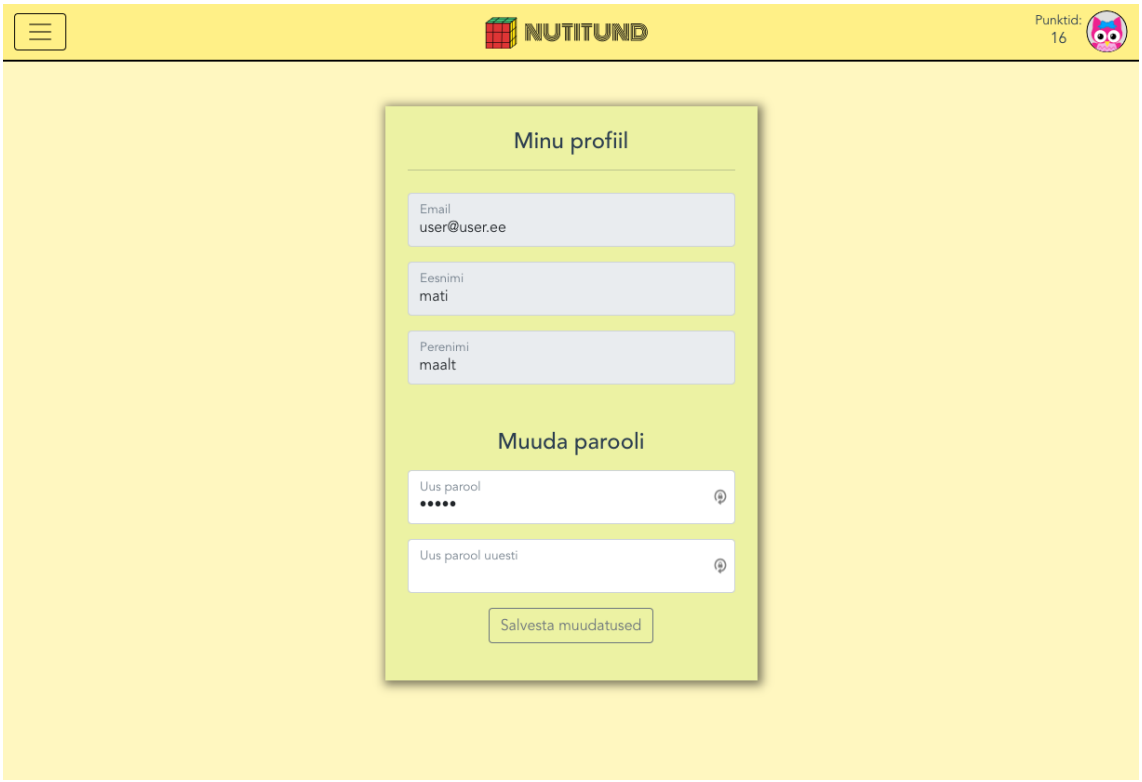
Lisa 3 – Kuvatõmmised loodud veebikeskkonnast




NUTITUND
Logi sisse
Logi sisse
Email
user@user.ee
Parool
•••••
Logi sisse
Grupi liige?
Ava
Uus kasutaja?
Loo konto


NUTITUND
Punktid:
16
Vali raskusaste
Loendamine
1. Õpime kokku lugema kuni arvuni 5.
Mitu pilve on kokku?
5
2
4
2. Õpime kokku lugema kuni arvuni 5.
3. Õpime kokku lugema kuni arvuni 5.
4. Õpime kokku lugema kuni arvuni 5.
Tubli! +1 Punkti


NUTITUND
Punktid:
15
Grupid
Et aidata kaasa laste andmete turvalisusele, on loomine ja kasutamine viis kuidas lapsed saavad keskkonda kasutada ilma, et peaksid jagama enda isikuandmeid.
Lapsed saavad sisse logida kasutajanime ja parooliga mille neile sisestate alljärgnevalt
Lastel on hiljem võimalus oma kasutaja seadetes parool endale sobivaks muuta.
Halda Gruppe
Tihased
Kasutajanimi parool
Kustuta Grupp
1 ulli *******
Eemalda
2 Kasutajanimi Parool
Lisa
Loo uus grupp


NUTITUND
Punktid:
16
Minu profiil
Email
user@user.ee
Eesnimi
mati
Perenimi
maalt
Muuda parooli
Uus parool
Uus parool uuesti
Salvesta muudatused