TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ergo Posti 179778IADB

Skaleeruv veebipõhine mänguline õppeplatvorm lastele

Bakalaureusetöö

Juhendaja: German Mumma
MSc

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: Ergo Posti

02.05.2021

Annotatsioon

Selle lõputöö eesmärk on luua skaleeruv multilinguaalne veebirakendus, mis võimaldab lastel oma silmaringi arendada mugavalt ja väikeste juppide kaupa. Taustana on esitatud põhjendused selle kohta, miks on mitmekesine õpe lastele varajases eas kasulik, samuti mitmesugused veebiõppe aspektid ning ka koroonapandeemiast tingitud veebiõppe meetodite ja skaleeruvuse vajadus. Rakenduse analüüsi eesmärk on esmalt välja tuua rakenduse nõuded ja seejärel võrrelda neid ka olemasolevate rakendustega. Samuti käib analüüsi alla kasutatavate tehnoloogiate valimine ja kirjeldamine. Analüüsi üks osa on ka rakenduse arhitektuuri planeerimine, et oleks võimalik täita rakenduse skaleeruvuse nõuet.

Lõputöö tulemus on minimaalne töötav rakendus, mis on skaleeruv ja mitmesse keelde tõlgitav. Praktilises osas on välja toodud selle rakenduse kasutajalood, millele järgneb kasutajaliidese disaini kirjeldamine ning ka huvitavamad osad koodist ja kasutatavate teenuste sidumisest. Kuna eesmärk on luua vaid minimaalne töötav rakendus, siis lõpetuseks on välja toodud ka olulisemad planeeritud edasiarendused.

Lõputöö on kirjutatud eesti keeles ning sisaldab 34-leheküljelist teksti, 6 peatükki ja 24 joonist.

Abstract

Scalable web-based gamified learning platform for kids

The goal of this thesis is to create a scalable multilingual web application that allows kids to widen their horizons comfortably and by going through small parts at a time. For background the main statements are about the necessity of diversified learning methods for children in young age, different aspects of online learning and also the need for online learning and scalability in such applications based on the recent corona pandemic. Target of the analysis in this thesis is to primarily bring out the requirements of the application and to compare these with already existing applications. In addition during the analysis the used technologies will be chosen and further described. As the last part of analysis is the planning of architecture so that the requirement of scalability is satisfied.

For the end result of this thesis is a minimal viable product or short for MVP, which is scalable and translatable into multiple languages. The practical part starts by defining the user stories for the application followed by designing of the application based on previously defined user stories. The next segment brings out the interesting parts of the code implementation and also binding of some services used by the application. As the objective of this thesis is to only implement the minimal viable product then as a follow up the main planned succeeding developments are represented.

The thesis is in Estonian and contains 34 pages of text, 6 chapters, 24 figures

Lühendite ja mõistete sõnastik

MVP	Minimum Viable Product, minimaalne töötav toode
SEO	Veebilehe optimeerimine otsingumootorite jaoks
REST	Representional State Transfer, Arvutisüsteemide vaheline standard veebis
API	Application Programming Interface, Kood, mis lubab andmete vahetust tarkvara süsteemide vahel
MVC	Model View Controller, Tarkvara disaini muster
XSS	Cross-Site Scripting, Murdskriptimine
NoSQL	Mitterelatsiooniline andmebaas
Main flow	Rakenduse põhivoog
Mobile-first	Rakenduse disainimise viis, kus esialgu disainitakse mobiilsetele seadetele
Plugin	Pistikprogramm
ID	Unikaalne identifikaator
React hook	React-il põhinev funktsioon, mis võimaldab kasutada olekut ja React-i teenuseid
Bearer Token	Pääsutõend
DDoS	Distributed Denial of Service, teenuse rõkestamise rünnak, mis toimub hajutatult
Toast	Kasutajaliidese esile tõstetud element
url	Veebiaadress

Sisukord

Jooniste loetelu

1 Sissejuhatus

Selle lõputöö raames valmis lastele silmaringi laiendamiseks mõeldud veebirakendus. Teema valimisel tekkis autoril esmane inspiratsioon Ameerika Ühendriikides Southwestern Advantage'i müüdavatest laste õpperaamatutest ning nendes pakutavatest mitmekesistest õppemeetoditest. Kuna koroonapandeemia tõttu tuli välja veebipõhise õppe lahenduste ning nende skaleeruvuse vajadus, siis otsustas autor proovida lõputöö kirjutamist sellel teemal.

Lõputöö eesmärgiks sai minimaalse töötava rakenduse loomine, mida oleks võimalik nii edasi arenda, skaleerida kui ka mitmesse keelde tõlkida. Seda selleks, et rakendusel oleks võimalik toetada võimalikult palju kasutajaid ja muutuvaid kasutajate koguseid.

Teises peatükis on autor kirjeldanud veebipõhise õppe tausta, toonud välja selle eeliseid ja ka puuduseid. Kolmandas peatükis on autor selgitanud esmalt rakenduse nõudeid, millele järgneb kasutatavate tehnoloogiate kirjeldamine ja valikute põhjendamine. Analüüsi viimases osas kirjeldab autor täpsemalt ka rakenduse planeeritavat arhitektuuri. Neljandas peatükis toob autor välja praktilise osa põhilisemad aspektid, alustades kasutajalugudest. Seejärel esitab autor rakenduses kasutatava disaini valikud ning ka rakenduse juurutamise huvitavamad kohad. Viiendas peatükis kirjeldab autor edasisi planeeritavaid arendusi.

2 Taust

Veebi laienev iseloom ja kättesaadavus on tekitanud veebipõhise õppe nõudluse kiire kasvu. Veebipõhine õpe on juba tunginud ka koolidesse ja ülikoolidesse. Mitmekesise õppe pakkumine lastele on näidanud, et sellel on hea mõju ka nende tulevikule. Samuti on koroonapandeemia esile toonud veebipõhise õppe vajalikkuse.

2.1 Õpe varajases lapsepõlves

Bakkeni, Browni ja Downingu [1] väitel vihjab varajane sekkumine laste mõistusesse, kehasse ja emotsioonidesse pikas perspektiivis mitmele eelisele laste elus. Varajase mitmekesise õppe kasulikkus avaldub õpilaste jaoks nii akadeemilisel kui ka sotsiaalsel tasandil ja samuti nende suhtumises. Akadeemiliselt on näha õpilaste paremaid tulemusi juba 3. klassist. Sotsiaalses aspektis tuleb välja laste asjalikum käitumine, suhete loomine, sotsiaalsed interaktsioonid ning ka emotsionaalne küpsus. Suhtumise puhul on nendel lastel rohkem koolis kohalviibimisi ja vähem puudumisi.

2.2 Veebipõhine õpe

Veebipõhine õpe toimub osaliselt või täielikult internetis. See on mitmekesine ja kohati ka keerukas nagu traditsiooniline klassiruumis toimuv õpe. Suurenenud kättesaadavus ja huvi distantsõppe vastu on palju suurendanud ka selle kasutamist koolides.

2.2.1 Eelised

Õpe, mis toimub interneti kaudu, on iseseisvam ning õpilased saavad panustada rohkem teemadesse, millega neil on abi vaja, ja vähem teemadesse, mis on neil selged [2].

Veebipõhise õppe paindlikkus pakub õpilastele ka võimalust õppida ajal ja kohas, mis sobib kõige paremini nende isiklike õpimeetoditega.

Tänapäeva maailmas muutuvad info ja teadmised kiiresti. Välja trükitud materjalidega on ainuke võimalus nende muutustega kaasas käia nende materjalide printimise kaudu. See

omakorda tähendab lisakulutusi, samal ajal kui veebimaterjalide uuendamine on tunduvalt soodsam ja kiirem [3].

2.2.2 Puudused

Üks põhilisi veebiõppe puudusi võrreldes traditsioonilise klassiruumi õppega on näost näkku suhtlemine. Nii õppe sotsiaalsed kui ka emotsionaalsed aspektid on olulised nagu ka tehnilised aspektid [3].

Veebipõhine õpe nõuab õpilastelt ka enesedistsipliini ja aja haldamise oskusi. Neid on vaja traditsioonilises klassiruumis, kuid veebipõhise õppe puhul tunduvalt rohkem, kuna õpetajapoolset juhtimist on vähem [3].

Mõned teemad pole ka veebipõhise õppe jaoks sobilikud. Keerulisemaid teemasid, mis vajavad füüsilist keskkonda, on lihtsam ja asjakohasem läbida kohapealses õppes [3].

2.3 Koroonapandeemia mõju e-õppele

Koroonapandeemiast tingitud õppeasutuste sulgemine on mõjutanud suurt osa inimkonnast nii suurte sotsiaalsete kui ka majanduslike kahjudega.

UNESCO [4] väitel mõjutatud aspektid:

- häiritud õpe. Koolid pakuvad olulist õpet ja koolide sulgemisega võetakse lastelt ära võimalused kasvamiseks ja arenemiseks. Taolised puudused tekivad kõige enam halvemas majanduslikus seisus õppijatel, kuna neil on vähem võimalusi end kooli kõrvalt harida;

- vanemad pole valmistunud distants- ega koduõppeks. Koolide sulgemisel muutub suur osa laste õpetamisest vanemate ülesandeks. Vanematel on tihtipeale aga vähe õpetamisoskuseid ja -vahendeid.

- distantsõppe loomise, haldamise ja arendamise probleemid. Koolide sulgemisel suureneb distantsõppe nõudlus tugevalt. See omakorda paneb olemasolevad distantsõppekeskkonnad suure koormuse alla. Klassiruumide liigutamine kodudesse suurtel skaaladel ja kiirustades tekitab suuri nii inimlikke kui ka tehnilisi katsumusi;

- majanduskulud. Töötavad vanemad puuduvad tõenäolisemalt töölt, et laste eest hoolt kanda. See avaldab mõju nii sissetuleku kui ka produktiivsuse langemisega.

3 Analüüs

Tarkvara arenduses on palju eri tehnikate kombinatsioone, mida vastava ülesande jaoks kasutada. Selles peatükis arutab autor nimetatud rakenduse nõudeid, tehnoloogiaid ja metoodikaid, mida rakenduse tegemisel kasutada.

3.1 Rakenduse nõuded

Eesmärk on luua ülemaailmselt skaleeruv mitmekeelne mänguline õppeplatvorm lastele, mida oleks ka lihtne edasi arendada. Õppeplatvorm, mis võimaldaks lastele alternatiivset õpet ning mille eesmärk oleks pigem silmaringi laiendamine kui kogu akadeemilise hariduse pakkumine. Selleks, et laste jaoks oleks õpe huvitav, on info kuvatud väikeste osadena kuni 50 sõnaga koos piltide, videote, animatsioonide ja audioga. Õpe toimub teemade ja nende kohta käivate moodulite kaupa. Igas moodulis on kuni 10 ülesannet. Esmalt kuvatakse kasutajale ülesanne ning selle õiget lahendust koos selgituse ja muu sisuga näeb kasutaja peale vastamist. Õige lahenduse puhul märgitakse kasutajal teema selgeks ja seda ülesannet enam ei kuvata. Vale lahenduse puhul läheb ülesanne mooduli lõppu ja seda kuvatakse kasutajale pärast teiste ülesannete täitmist uuesti.

Selleks, et esialgu oleks sisu võimalikult korrektne ja kvaliteetne, on rakendus jaotatud kahte osasse: kasutajapoolne rakendus, mille ülesanne on kogu enne kirjeldatud funktsionaalsuse pakkumine kasutajale, ja halduripoolne rakendus, mille ülesanne on võimaldada rakenduse sisu lisamist ja haldamist.

3.1.1 Funktsionaalsed nõuded

Loodavas rakenduses on planeeritud järgnevad funktsionaalsed nõuded:

- Halduri süsteemis peab saama lisada ja hallata teemasid.
- Halduri süsteemis peab saama lisada ja hallata õppemooduleid.
- Halduri süsteemis peab saama lisada ja hallata küsimusi.
- Halduri süsteemis peab saama lisada ja hallata küsimuste vastusevariante.
- Halduri süsteemis peab saama lisada ja hallata mitmesuguse sisuga seotud faile.

- Halduri süsteemis peab saama lisada ja hallata dünaamilise ja ka staatilise sisu tõlkeid.
- Kasutaja süsteemis peab saama kasutajana registreerida ja sisse logida.
- Kasutaja süsteemis peab saama sisu kuvada.
- Kasutaja süsteemis peab saama kasutaja progressi salvestada.

3.1.2 Sarnased rakendused

Nagu ka varem mainitud, siis e-õppe rakendusi on mitu. Kuigi allpool mainitud rakendused pole selles lõputöös tehtud rakenduse otsesed konkurendid, siis osaliselt on nende lahendused siiski sarnased.

Duolingo

Kuigi Duolingol on funktsionaalsuselt väga sarnane lahendus praeguses lõputöös tehtava rakendusega, siis selle fookus on spetsiifiliselt keeleõppel. Lõputöös tehtava rakenduse fookus on pigem silmaringi laiendamisel eri teemade kohta.

National Geographic Kidz

Sisu poolest kõige sarnasem rakendus, kuigi funktsionaalsus on veidi erinev. Kasutajal puuduvad võimalused oma progressi jälgimiseks ning platvorm on ainult inglise keeles.

Opiq ja E-koolikott

Üks levinumaid eestikeelseid õppeplatvorme. Kuigi see on kvaliteetse ja koguka sisuga, siis eesmärk on pigem just hariduse omandamine. Nagu enne mainitud, siis lõputööna tehtava rakenduse eesmärk on just laste silmaringi laiendamine, mitte suuremas pildis akadeemilise hariduse pakkumine.

3.1.3 Skaleeruvus

Rakenduse skaleeruvus näitab, kui palju samaaegseid päringuid on ta võimeline efektiivselt toetama. Punkt, kus rakendus ei ole enam võimeline lisapäringuid efektiivselt käsitlema, on tema skaleeruvuse limiit. See tekib siis, kui riistvaraline ressurss on otsas ja on vaja kasutada kas uut või mitut masinat [5].

Kuna seoses koroonapandeemiaga on eriti esile tulnud ka olukorrad, kus veebipõhiste õppeplatvormide kasutatavus tõuseb tugevalt, siis võttis autor selles lõputöös eesmärgi teha rakendus, mida on lihtne suureneva koormuse puhul skaleerida.

Vertikaalne skaleerumine on ühe masina võimendamine, mis on tihtipeale lihtsam ja ka odavam, kuna rakenduse loogika ei pea muutuma ja riistvara hind on järjest odavam. Ühte masinat ei saa aga lõpmatuseni võimsamaks skaleeruda ja mingist hetkest tekivad piirangud. Alternatiiv on kasutada horisontaalset skaleeruvust, mis tähendab rakenduse majutamist mitme masina peal [5].

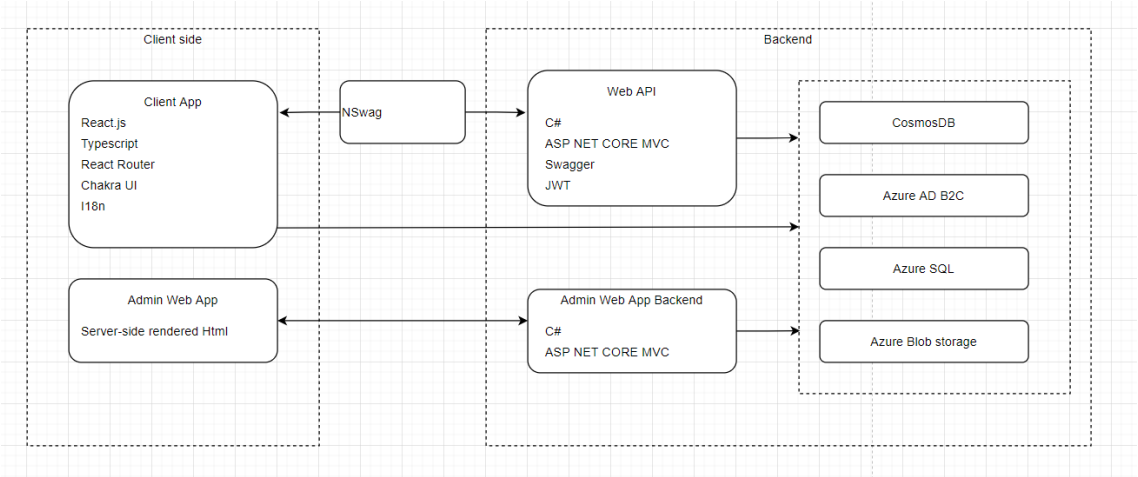
3.2 Pilveteenused ja rakenduse majutamine

Pilveandmetöötlus on erinevate teenuste pakkumine läbi interneti. Pilveteenuste kasutamine rakenduse arendamisel võimaldab vähendada kulusid, suurendada nii produktiivsust, kiirust ja efektiivsust [6].

Kuna rakendus peab olema lihtsasti skaleeruv, edasi arendatav ja ülemaailmselt kättesaadav, siis rakenduse arendamiseks võttis autor kasutusele erinevaid pilveteenuseid. Samuti lähtus autor ka teenuste üleüldise haldamise mugavusest. Selle põhjal otsustas autor nii veebiserverid kui ka veebilehe staatilist osa majutada Azure keskkonnas, kus asuvad ka teised teenused, millest rakendus sõltub.

3.3 Kasutatavad tehnoloogiad

Kasutatavate tehnoloogiate valimisel võttis autor arvesse mitut aspekti. Need olid tehnoloogiate populaarsus, nende sobilikkus üksteisega ning ka isiklikud eelistused. Et oleks ülevaade ka rakenduse loomisel kasutatud tehnoloogiatest, siis tegi autor nendest joonise (Joonis 1).



Joonis 1. Kasutatud tehnoloogiad.

3.3.1 Tugev tüüpimine

Programmeerimiskeele üks põhilisi vaadatud omadusi on keele turvalisus, kus kompilaator on püüdnud kinni iga katse andmeid valesti interpreteerida. Suure osa sellest turvalisusest saab täita tugevalt tüüpimisega. Tugevalt tüüpides iga muutuja ja avaldise tüübi saab teada, kui vaadata programmi ilma seda tööle panemata. Muutujat saavad kasutada ainult meetodid, mis tunnustavad tema tüüpi [7].

Mõned vead tekivad arusaama puudumisest, teised on loogilised vead, mida põhjustavad ebapiisav mõtlemine ja disain, ning mõned on lihtsalt kirjavead. Igal juhul võib nende vigade leidmine kompileerimisel säästa palju aega. Turvalisus ja tugevalt tüüpimine teevad võimalikuks paljude vigade varajase avastamise [7].

Kuna tehtav rakendus peab olema skaleeruv, pidevalt edasi arendatav ning samas ka võimalikult väheste vigadega, siis tehnoloogiate valimisel on eelistatud tugevalt tüüpimine või selle võimalik lisamine.

3.3.2 C# ja ASP.NET Core MVC

C# on mitme paradigmaga programmeerimiskeel, mis hõlmab tugevalt tüüpimist, funktionaalsust, klassipõhist objektorienteeritust. Keele arendaja ja haldaja on Microsoft koos .NET initsiatiiviga.

ASP.NET Core MVC on veebirakenduste arendamise raamistik. Kasutusel on MVC arhitektuur ja agiilse arenduse tehnika [8]. Põhiline põhjus selle kasutamiseks rakenduse

loomisel on, et ASP.NET Core MVC töötab .NET Core peal ehk rakendust saab jooksutada mitmel platvormil, mitte ainult Windowsil.

3.3.3 React ja Typescript

React on avatud lähtekoodiga Javascript teek, mida kasutatakse kasutajaliidese loomiseks. Reactil põhinev rakendus koosneb kapsuleeritud vaate komponentidest. Komponente saab omavahel komponeerida, et luua keerulisi kasutujaliideseid [9].

Typescript on avatud lähtekoodiga Javascripti laiendus, mis võimaldab staatilist tugevalt tüüpimist. Typescript abil on võimalik kirjeldada objektide sisu, paremat dokumentatsiooni ja koodi korrektsuse valideerimist. Typescripti kood transformeeritakse Javascriptiks Typescript kompilaatori või Babeli kaudu [10].

Eesmärk on luua mänguline õppeplatvorm ning seetõttu on rakenduses olevad kasutajaliidesed väga mitmekesised ja koosnevad paljudest osadest. Selle saavutamiseks on ka mitmesuguseid olemasolevaid tehnoloogiaid, nagu Vue.js, Angular ja Aurelia, aga autori kogemuse põhjal võeti kasutusele React. Samuti lubab React suurt osa koodist kirjutada Typescriptiga, mis võimaldab ka kliendipoolse rakenduse kirjutada tugevalt tüübitult.

3.3.4 OpenAPI ja NSwag

OpenAPI standard defineerib keele-agnostilise liidese RESTful APIle, mis võimaldab nii inimestel kui ka arvutitel avastada ja mõista teenuse võimalusi ilma lähtekoodile ligipääsuta, dokumentatsioonita või võrgu liikluse inspektsioonita. OpenAPI definitsiooni saab kasutada nii dokumentatsiooni kui ka serverite ja klientide koodi genereerimiseks ning seda mitmes programmeerimiskeeles [11].

NSwag on tehnoloogia, mis kombineerib OpenAPI genereerimise ja OpenAPI põhjal kliendi genereerimise [12]. Seda funktsionaalsust saab ära kasutada rakenduses serveri ja kliendi rakenduse ühendamiseks tugevalt tüübitud viisil.

3.3.5 Azure Active Directory B2C

Azure Active Directory B2C pakub ärilt kasutajale identiteeti teenusena. Kasutajad saavad valida eelistatud sotsiaalse, ettevõttepõhise või lokaalse identiteedi, et saada ühekordne sisselogimise ligipääs rakendusele [13].

Mingist hetkest tekivad paljudesse veebirakendustesse mitmesugused potentsiaalsed turvavead. OWASP [14] andmetel kuuluvad levinumate turvavigade hulka näiteks süstimisrünne, autentimise ja sessionihalduse vead, tundliku info kaitseta jätmine ja ka murdskriptimine. Selleks, et vähendada turvariske seoses kasutajate andmetega, otsustas autor kasutajate haldamiseks kasutusele võtta välise teenuse Azure Active Directory B2C.

3.3.6 Azure Cosmos DB

Azure Cosmos DB on üleni hallatud NoSQL andmebaas, mis pakub mitmeregioonilist andmejaotust üle maailma. Samuti pakub teenus 99,99% teenusetaseme lepingut ning mitme regioonilise kirjutamise puhul 99,999% teenusetaseme lepingut [15].

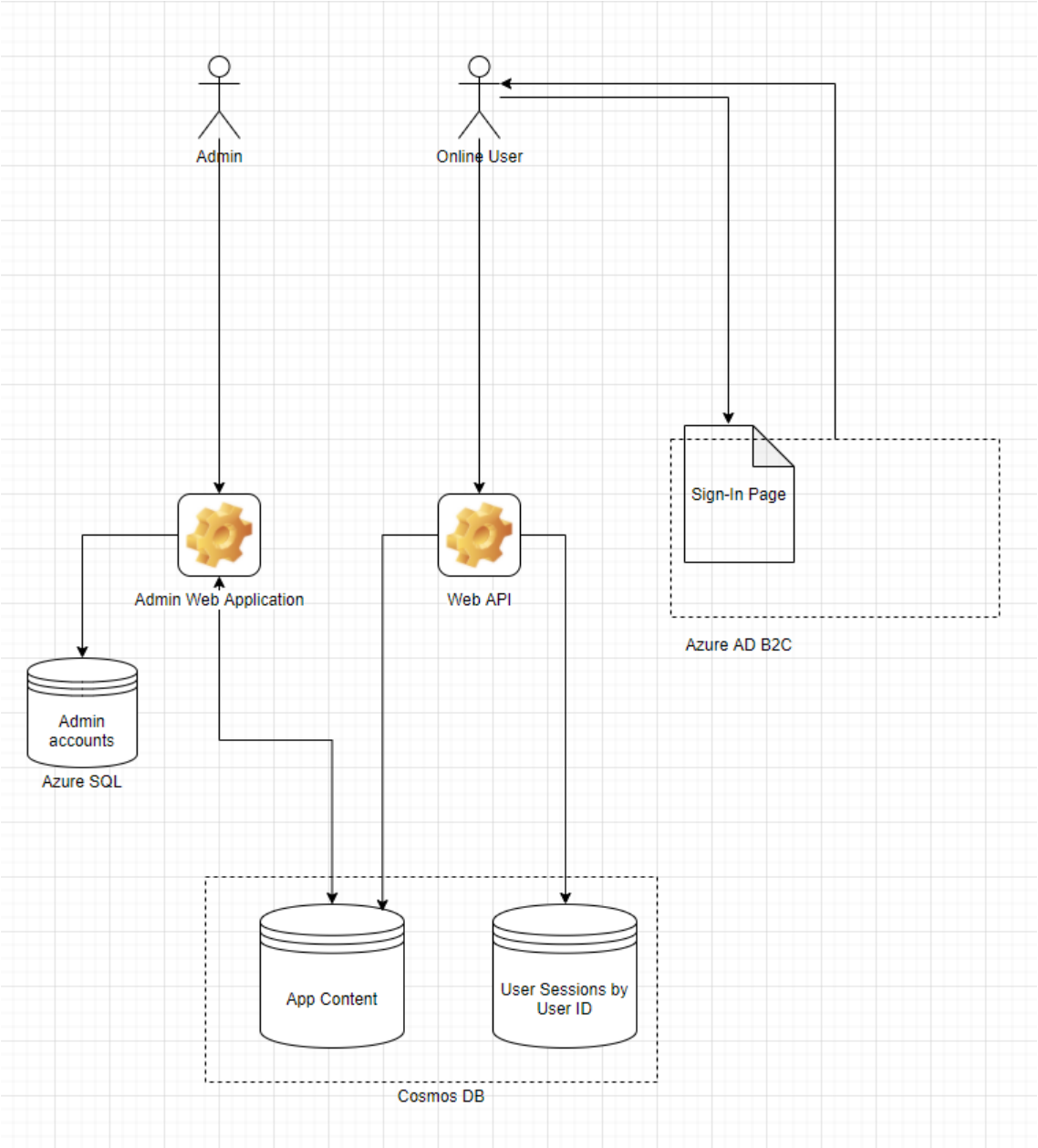
Kuna lõputöös tehtud rakendus peab olema horisontaalselt skaleeruv, ülemaailmselt kättesaadav ja ei vaja samas tugevaid relatsioone andmete vahel, siis NoSQL ja spetsiifiliselt Cosmos DB on sobilikud tehnoloogiad teenuse andmete hoidmiseks.

3.4 Arhitektuur

Arhitektuuri loomiseks on põhiliselt arvestatud rakenduse skaleeruvat ja ülemaailmset loomust. Selleks on vaja arvestada, et teenus on majutatud mitme serveri ja andmebaasi peale.

3.4.1 Kasutaja haldus

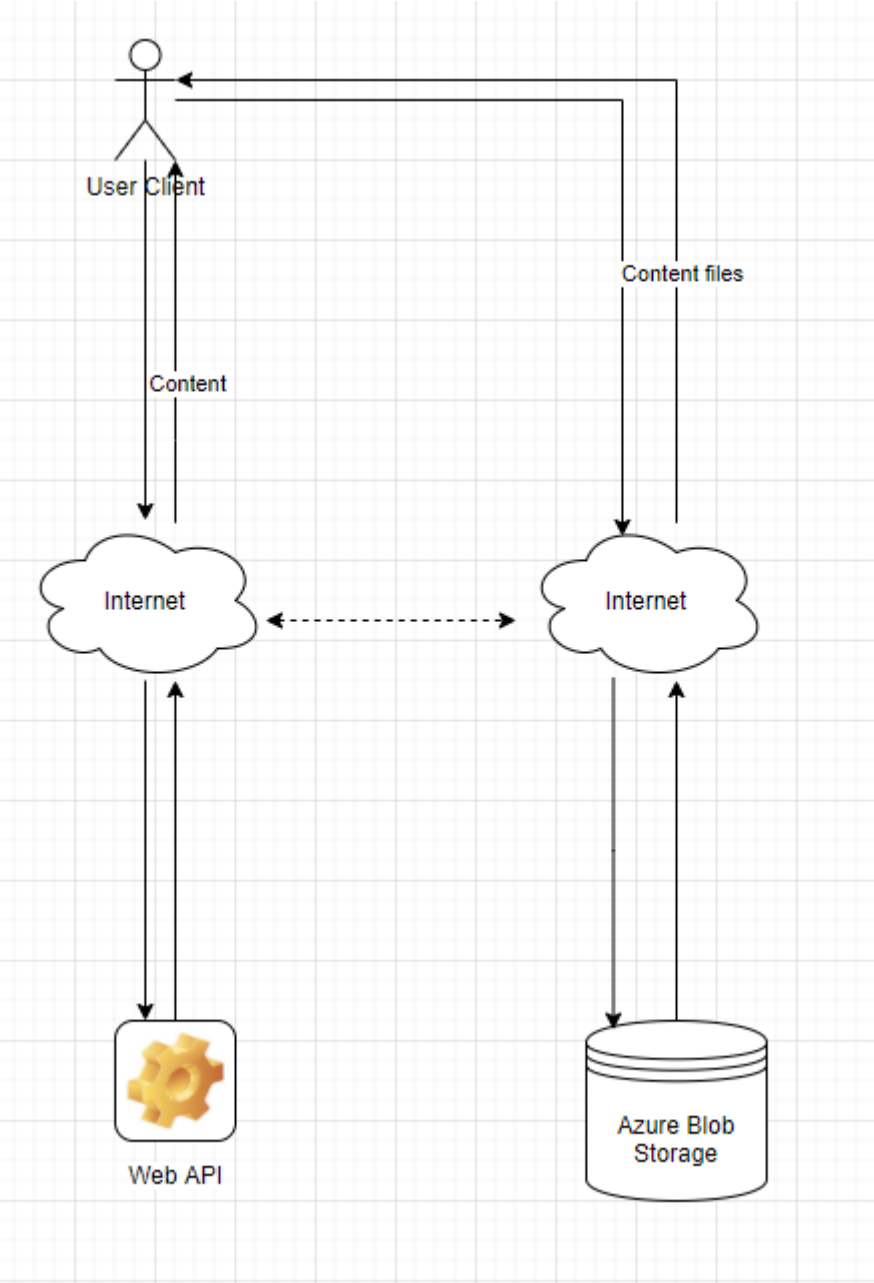
Kuna tavakasutaja rakendus peab olema võimalikult kiire ja interaktiivne, kuid samas administraatoril taolisi eelisi vaja ei ole, siis on kasutaja rakendus administreerimise osast täiesti eraldi. Kasutaja sisselogimine toimub rakendusest eraldi teenuse Azure AD B2C sisselogimislehe kaudu, mis taustal tagastab vajalikud sisselogimisandmed rakendusele. Rakenduse andmebaasis on kasutaja progressi hoidmiseks ja tagastamiseks vaja ainult kasutaja kordumatut tunnust. Kui kasutaja progressi peab kasutajapoolne rakendus saama muuta, siis rakenduse sisu tohib see ainult lugeda. Selle tagamiseks on kasutajate progressid ja rakenduse sisu andmebaasis eraldi konteinerites, millel on vastavad ligipääsuõigused. Rakenduse sisu saab muuta ainult administraatori veebirakenduse kaudu.



Joonis 2. Kasutajate halduse diagramm.

3.4.2 Kasutaja voog sisu vaatamisel

Kuna Cosmos DB on mõeldud pigem väheste andmete kiireks edasi-tagasi liigutamiseks, siis pole see sobilik koht, kus hoida pilte ja muid faile, mis võtavad üsna palju ruumi. Rakenduse failide haldamiseks ja hoidmiseks on kasutusel Azure Blob Storage. Kasutajale sisu kuvamisel päritakse kõigepealt sisu ja siis sisus viidatud failid eraldi Azure Blob Storage kaudu (Joonis 3).



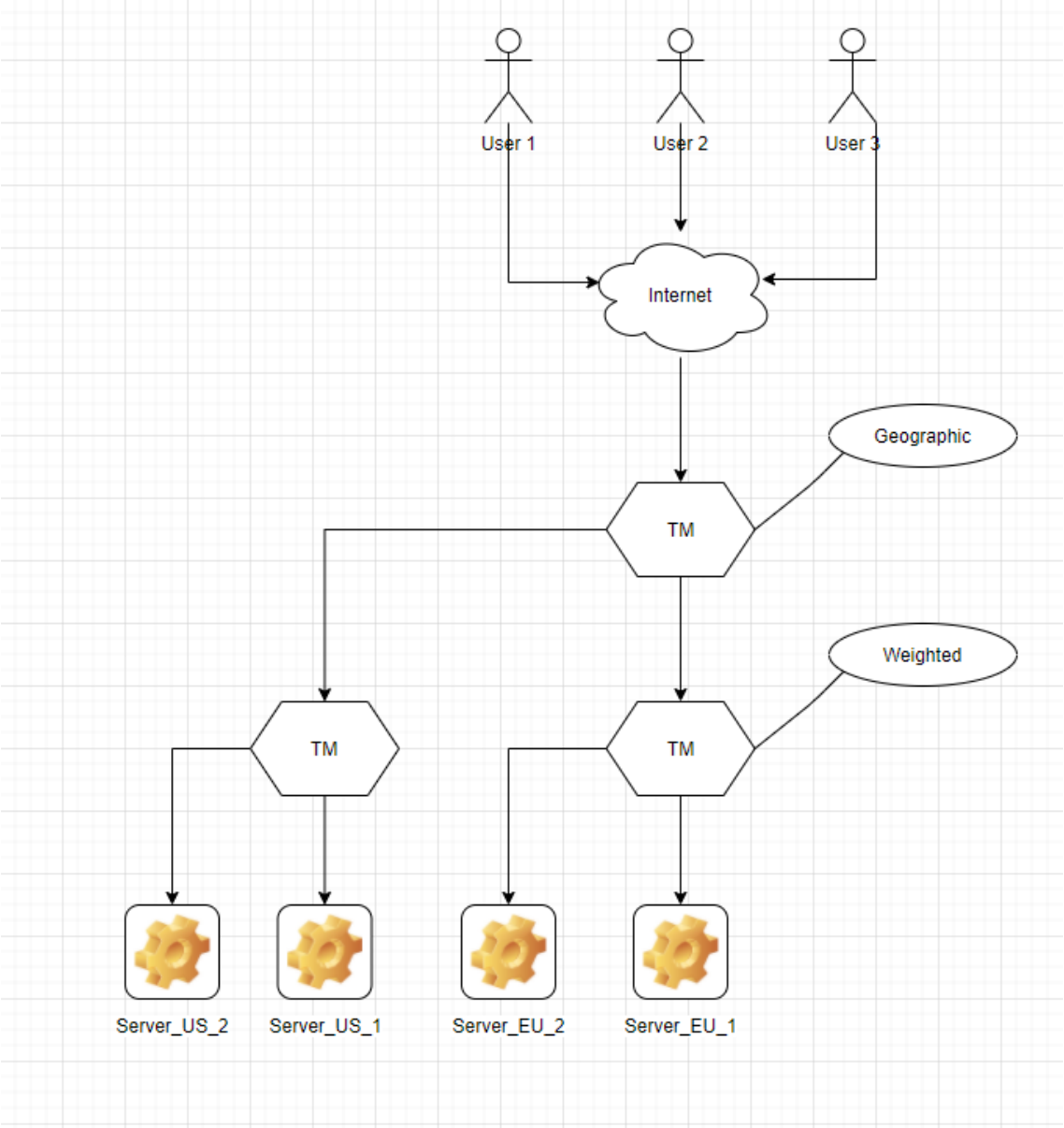
Joonis 3. Kasutajate sisu vaatamise voo diagramm.

3.4.3 Koormuse tasakaalustamine

Koormuse tasakaalustamine tähendab efektiivselt sissetulevate päringute jaotamist üle mitme serveri. Modernsed veebirakendused peavad täitma tuhandeid kui mitte miljoneid päringuid kasutajatelt või klientidelt. Samuti peavad nad seda tegema kiirel ja usaldusväärsel viisil. Selleks, et sellise suure koormusega vastu pidada tootlikul viisil, on enamasti parim praktika lisada uusi servereid ehk skaleeruda horisontaalselt [16].

Koormuse tasakaalustaja ülesanne on olla vahepealne etapp kasutajate päringute ja serverite vahel. Tasakaalustaja saadab kasutaja päringud edasi serveritele, mis on võimelised neid päringuid teenindama, ning kindlustab, et ükski server ei oleks üle koormatud [16].

Siinse rakenduse puhul on autor tasakaalu jaotamiseks valinud 2 etappi, millest esimene on geograafiline, mis jaotab kasutajate päringuid geograafilise asukoha järgi, et päringu latentsus oleks võimalikult väike. Teine etapp on päringute jaotamine koormuse järgi, mis jaotab päringuid serverite võimekuse põhjal (Joonis 4).

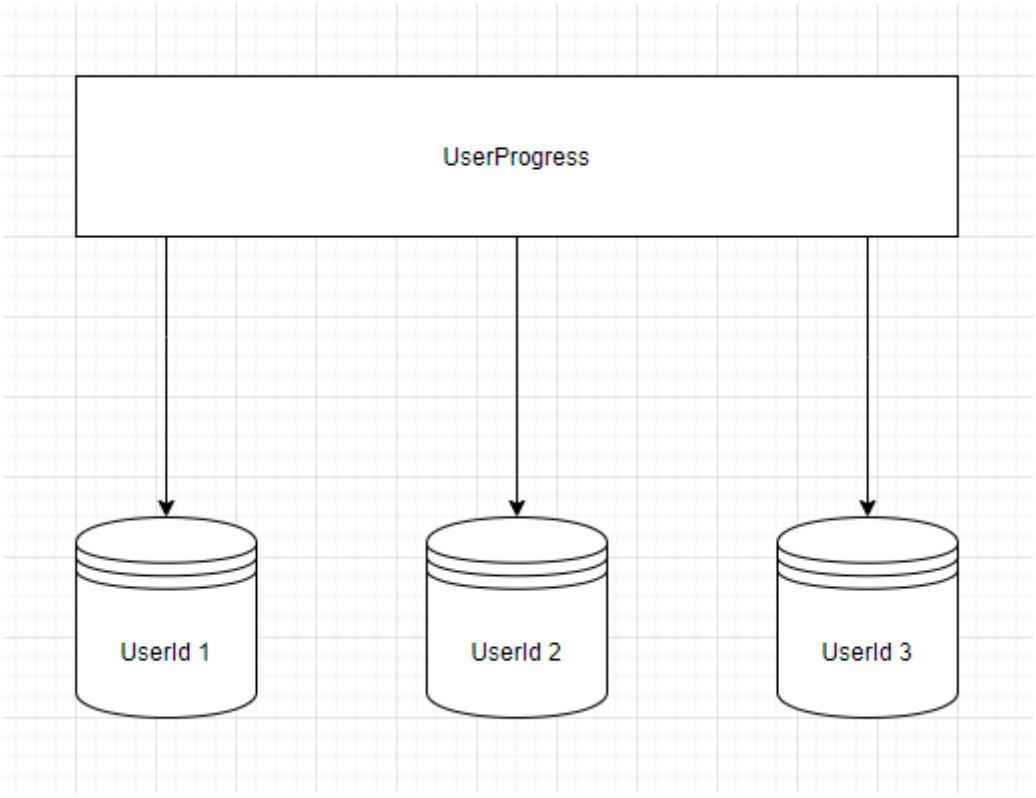


Joonis 4. Koormuse tasakaalustamise diagramm.

3.4.4 Eraldamine

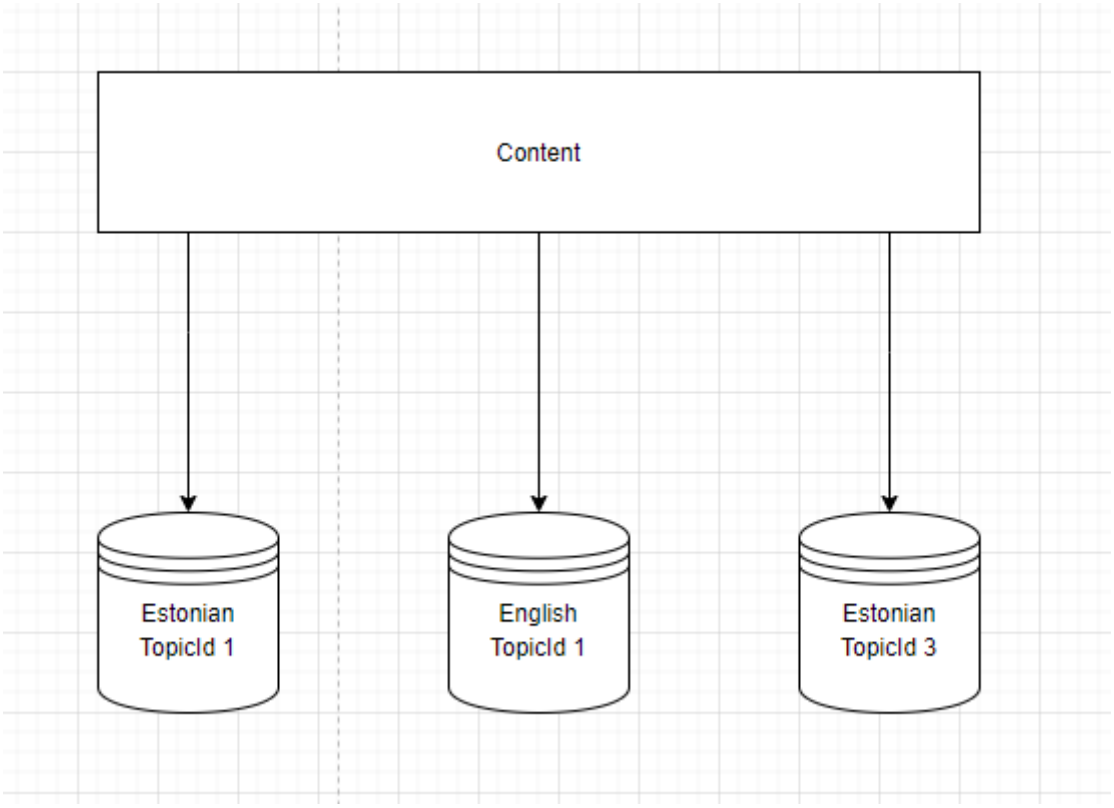
Suurte andmekogude või suure läbilaskevõimega süsteemide puhul ei ole alati võimalik kõiki andmeid ühes andmebaasis hoida. Sellistel juhtudel on vaja andmed jaotada eraldi *shared-nothing* osadeks ehk osadeks, mis ei pea omavahel seotud olema. Nii saab andmebaasi jagada üle mitme ketta ja koormuse jaotada üle mitme protsessori [17].

Lõputöös tehtava rakenduse puhul on põhiliselt kasutaja andmed need, mis dünaamiliselt muutuvad. Kuna kasutajatel puuduvad omavahelised seosed, siis saab kasutajad eraldada kasutaja kordumatu tunnuse järgi (Joonis 5).



Joonis 5. Kasutaja progressi eraldamise diagramm.

Kuigi esialgsel rakendusel on sisu üsna vähe, siis pikemas perspektiivis tuleb kindlasti arvestada ka sisu kasvuga. Kuigi ühe teema sisse jääb enam-vähem samas suurusjärgus õppemooduleid, küsimusi ja vastuseid, siis teemasid võib kasvata lõpmatuseni. Samuti kuna platvorm on mitmekeelne, siis iga keele lisamine omakorda suurendab sisu. Sellest tingitult kasvab sisu ruumivajadus N*M keerukusel, kus N on teemade arv ja M on keskkonna keelte arv. Kuigi teemade sisuvaheline seos on tugev, siis teemade omavaheline seos ja ka keelte omavaheline seos on nõrk. Sel põhjusel sobivad nii keel kui ka teema eraldusvõtmeks (Joonis 6).



Joonis 6. Dünaamilise sisu eraldamise diagramm.

4 Praktiline osa

Selles peatükis on kirjeldatud lõputöö implementeerimise spetsiifilised osad. Praktilises osas lähtus autor eesmärgist luua minimaalne kasutatav rakendus ehk MVP. Peatüki alguses on kirjeldatud kasutajalugusid, mille põhjal tegi autor disaini ja ka lõpliku rakenduse.

4.1 Kasutajalood

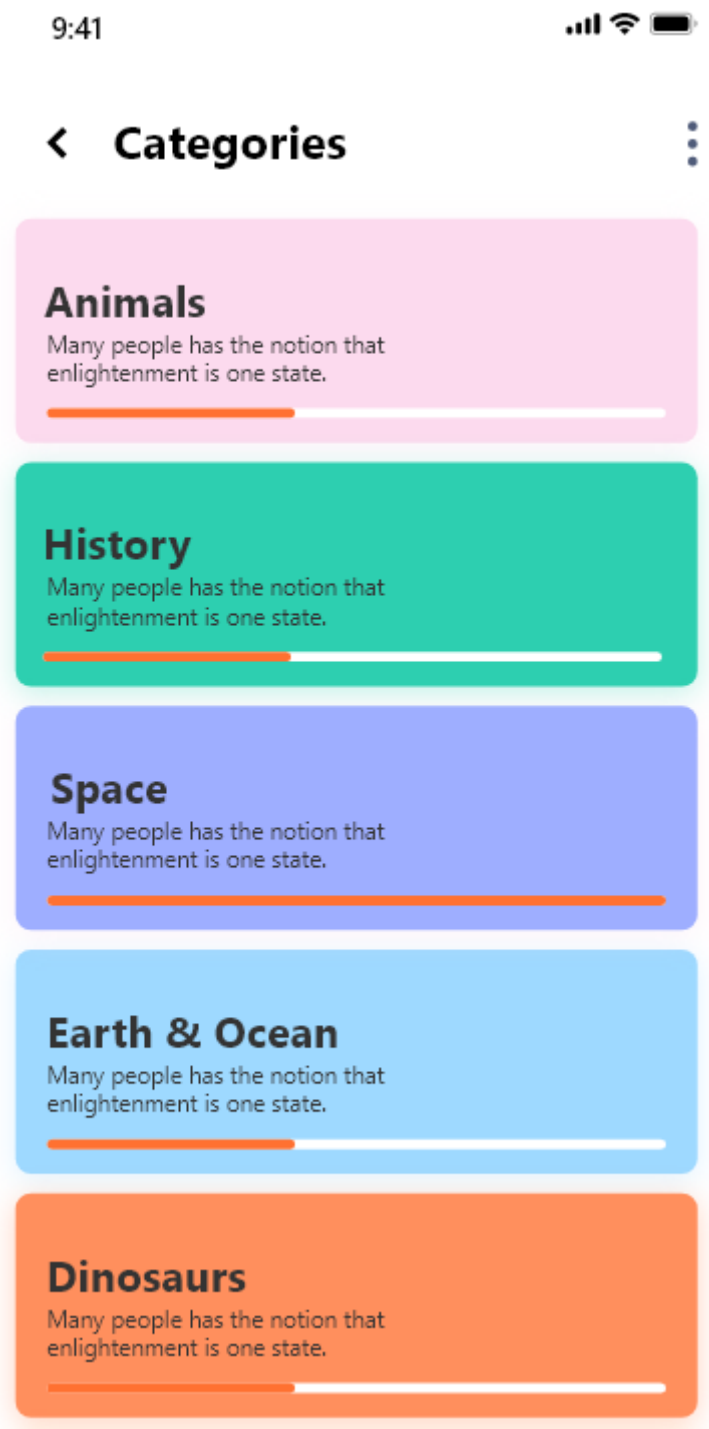
Loodavas rakenduses on realiseeritud järgnevad kasutuslood:

- Teenuse kasutajana kategooriate vaates tahan näha võimalikke valitavaid kategooriaid.
- Teenuse kasutajana kategooriate vaates tahan näha oma progressi kindla kategooria raames.
- Teenuse kasutajana kategooria õppemoodulite vaates tahan valida endale meelepärast õppemoodulit.
- Teenuse kasutajana kategooria õppemoodulite vaates tahan näha, millised õppemoodulid olen varem läbinud.
- Teenuse kasutajana õppemooduli vaatesse minnes tahan näha nimetatud ülesannet.
- Teenuse kasutajana õppemooduli küsimustiku ülesande vaates tahan näha, mitu küsimust on kokku ja mitu õigesti vastatud.
- Teenuse kasutajana õppemooduli küsimustiku ülesande vaates valikvastuse valimisel tahan näha õiget vastust, selle selgitust ja nuppu järgmise küsimuse peale minekuks.
- Teenuse kasutajana õppemooduli küsimustiku ülesande vaates, kui kõik küsimused on vastatud, tahan näha nuppu, mis võimaldab minna järgmise õppemooduli peale.

4.2 Disain

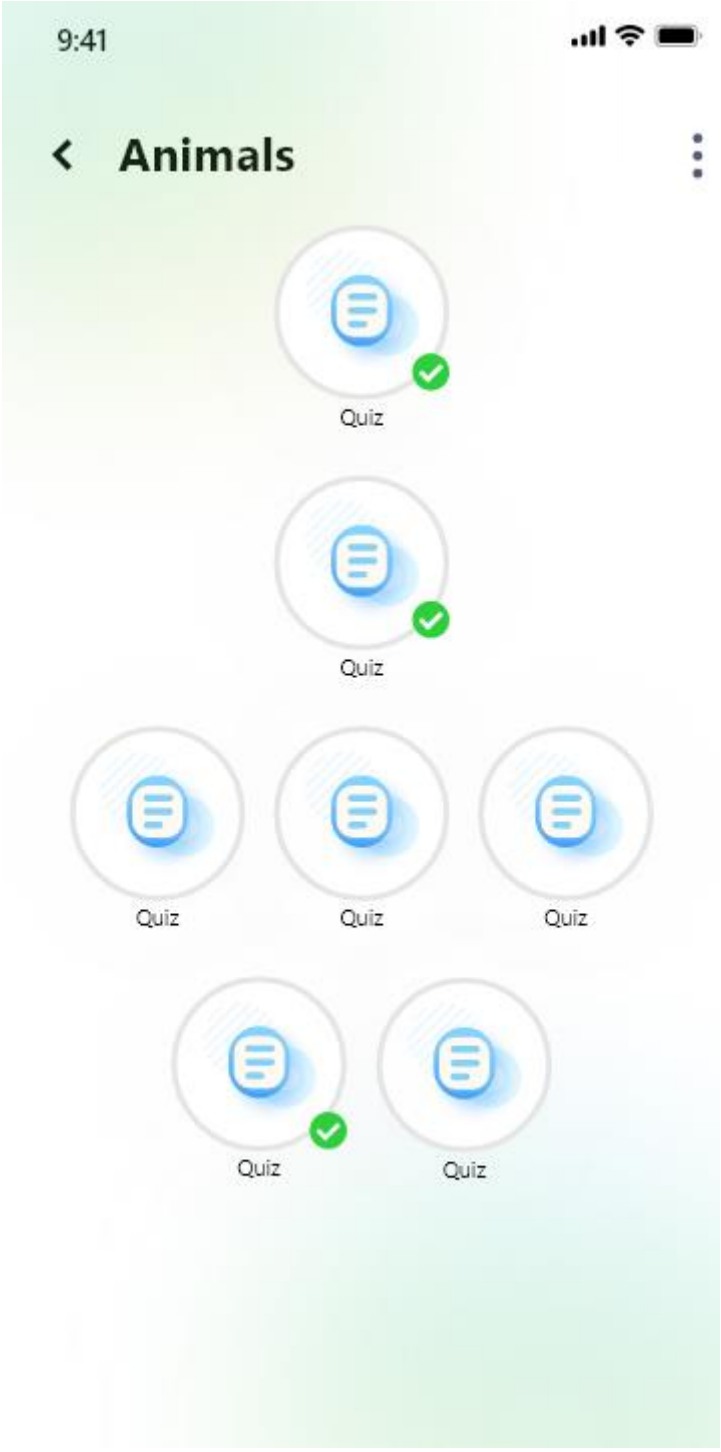
Selles peatükis on esitatud kasutajapoolse rakenduse MVP/Main flow disainitud komponendid. Disainimisel on autor lähtunud minimalistlikust disainist ning ka Mobile-first meetodist, et rakendust saaks vaikimisi kasutada võimalikult paljudel seadmetel.

Kategooriate vaate eesmärk on kuvada kasutajale pakutavad kategooriaid, millele vajutades saab kasutaja minna vastava kategooria vaatesse. Kategooria peal on ka kuvatud see, kui palju sellest kategooriast on kasutaja läbi teinud (Joonis 7).



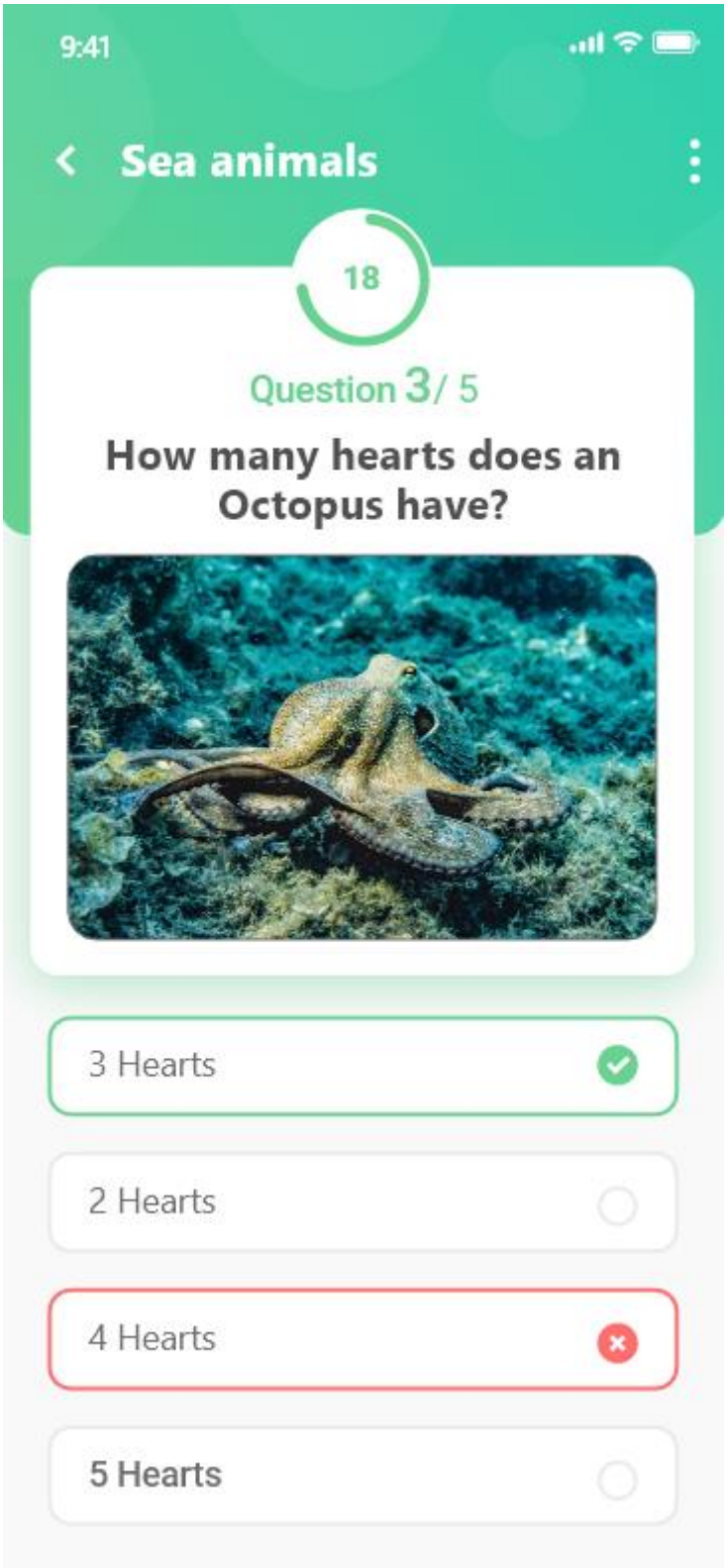
Joonis 7. Kategooria vaate disain.

Õppemoodulite vaates on kasutajale kuvatud varem valitud kategooria kohta käivad õppemoodulid. Õppemoodulile vajutades läheb kasutaja vastavasse küsimustiku vaatesse. Iga läbitud mooduli peal on ka linnukesega märgitud see, kui moodul on läbitud (Joonis 8).



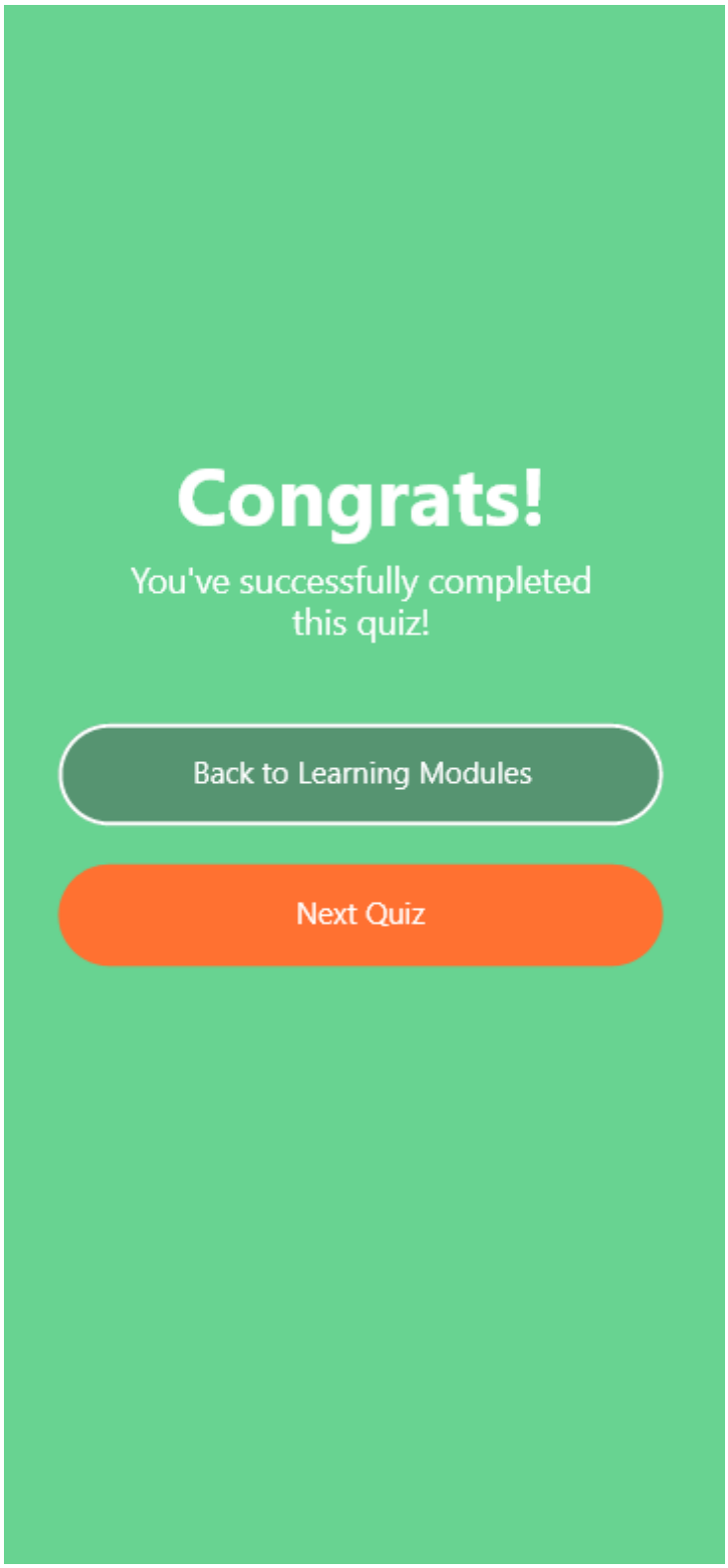
Joonis 8. Õppemoodulite vaate disain.

Ülesannete vaates kuvatakse kasutajale küsimus ja küsimuse juurde lisatud pilt koos valikvastustega. Vastuse valimisel kuvatakse kasutajale, kas vastus oli õige, ja vale vastuse puhul kuvatakse õige. Samuti kuvatakse kasutajale peale vastamist lühidalt õige vastuse selgitus ning järgmise küsimuse juurde mineku nupp. Kui kasutaja vastab küsimusele valesti, siis küsimus läheb küsimustiku lõppu ja see kuvatakse kasutajale uuesti. See kordub seni, kuni iga küsimus küsimustikus on õigesti vastatud (Joonis 9).



Joonis 9. Küsimustiku vaate disain.

Kui ülesanded on läbitud, siis kuvatakse kasutajale õppemooduli lõpetamise vaade, kus kasutajal on võimalus minna otse järgmise õppemooduli peale või minna tagasi kategooria vaatesse (Joonis 10).



Joonis 10. Küsimustiku lõpetamise vaate disain.

4.3 Kood

Selles peatükis on autor esitanud osa kohti praktilise osa koodist. Nende koodijuppide valikul lähtus autor põhiliselt punktidest, mis olid keerukamad ja erinesid mingil määral traditsioonilistest arendusmeetoditest.

4.3.1 I18n seadistamine ja kasutamine

Tehtud rakenduse üks nõue on, et seda oleks võimalikult paljude inimeste jaoks mugav kasutada. Selle täitmiseks oli peale infrastruktuurile vaja lisada ka võimalus sisu tõlkimiseks. Et seda saavutada kasutajapoolses rakenduses, võttis autor kasutusele tehnoloogiad i18n ja react-i18n. Selles peatükis on esitatud nende tehnoloogiate seadistamine nii kasutajapoolses kui ka serveripoolses rakenduses ja näide nende kasutamise kohta.

Nimetatud näites on välja toodud, kuidas on kasutajapoolses rakenduses tehnoloogia seadistatud. Rakenduse toetatud keeled on loetletud konfiguratsioonis keelte koodide jadana. Arvestades neid keeli ja kasutaja brauseri keelt, leiab i18n plugin i18next-browser-languagedetector kõige sobilikuma keele ning laeb selle keele ressursid serverist seadistatud aadressilt (Joonis 11).

```
import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';
import Backend from 'i18next-http-backend';
import LanguageDetector from 'i18next-browser-languagedetector';
import config from './config';

i18n
  .use(Backend)
  .use(initReactI18next)
  .use(LanguageDetector)
  .init({
    fallbackLng: config.fallBackLanguageCode,
    debug: config.isDebug,
    react: {
        useSuspense: false
    },
    interpolation: {
      escapeValue: false,
    },
    backend: {
        loadPath: `${config.apiBaseUrl}resources/Export/ByLanguageCode?langua
geCode={{lng}}`,
    },
    supportedLngs: config.supportedLanguages,
  });

export default i18n;
```

Joonis 11. I18n konfigureerimise kood.

Allärgnevas näites on välja toodud serveripoolne API controller, mis tagastab vastavad tõlkeressursid kasutajapoolsele rakendusele. Kontroller tagastab tõlkeressursside kollektsiooni keele unikaalse identifikaatori või keelekoodi põhjal (Joonis 12). Tõlke ressurss koosneb IDst, keele-IDst, võtmest ja väärtusest (Joonis 13).

```
[Route("resources/[controller]")]
[ApiController]
public class ExportController : ControllerBase
{
    private readonly IRepository<TranslationResource> _translationResourcesRe
pository;
    private readonly IRepository<Language> _languageRepository;

    public ExportController(IRepository<TranslationResource> translationResou
rcesRepository, IRepository<Language> languageRepository)
    {
        _translationResourcesRepository = translationResourcesRepository;
        _languageRepository = languageRepository;
    }

    [Route("ById")]
    [HttpGet]
    [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Dictionary<s
tring, string>))]
    [ProducesResponseType(StatusCodes.Status404NotFound)]
    public async Task<IActionResult> GetTranslationResourcesById(Guid languag
eId)
    {
        var resources = await _translationResourcesRepository.TableNoTracking
.Where(x => x.LanguageId == languageId).ToListAsync();

        return Ok(resources.ToDictionary(x => x.Key, x => x.Value));
    }

    [Route("ByLanguageCode")]
    [HttpGet]
    [ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Dictionary<s
tring, string>))]
    [ProducesResponseType(StatusCodes.Status404NotFound)]
    public async Task<IActionResult> GetTranslationResourcesByLanguageCode(st
ring languageCode)
    {
        var language = await _languageRepository.TableNoTracking.SingleOrDefa
ultAsync(x => x.LanguageCode == languageCode);

        return await GetTranslationResourcesById(language.Id);
    }
}
```

Joonis 12. Keeleressursside eksportimise kood.

```
public class TranslationResource : BaseEntity
{
	public Guid LanguageId { get; set; }
	public string Key { get; set; }
	public string Value { get; set; }
}
```

Joonis 13. Tõlkeressursi olemi kood.

Tõlgitud tekstide kasutamiseks töötab autor rakenduses react-i18next „useTranslation“ React Hookiga, mis võtab sisendiks tõlkeressursi võtme ja tagastab tõlgitud väärtuse seadistatud keele järgi (Joonis 14).

```
import { useTranslation } from "react-i18next";

const translationExample = () => {
	const {t, i18n} = useTranslation();

	return (
		<div>
			{t("translationKey")}
		</div>
	)
};
```

Joonis 14. I18n kasutamine rakenduses kood.

4.3.2 Kasutaja taustal sisse logimine ja Bearer Tokeni hoidmine

Kuna rakendusse sisse logimine käib välise teenuse (Azure AD B2C) kaudu, siis kasutaja autentimise andmete värskendamine ja hoidmine on natukene keerulisem kui tavaliste rakenduste puhul. Selleks lõi autor vastava React Hooki, mis kasutaja värske Bearer Token’i olemasolu puhul tagastab selle. Juhul kui Bearer Token on aegumas, proovib React Hook seda tagaplaanil uuendada, et tagastada see uuendatult. (Joonis 15).

```
import { useAccount, useMsal } from "@azure/msal-react";
import { useToast } from "@chakra-ui/react";
import { useState } from "react";
import { useTranslation } from "react-i18next";
import { loginRegisterRequest } from "../config/authConfig";

const useAccountBearerToken = () => {
    const [bearerToken, setBearerToken] = useState<string>(null);
    const { t } = useTranslation();
    const { instance, accounts, inProgress } = useMsal();
    const account = useAccount(accounts[0] || {});
    const toast = useToast();

    if(account) {
        instance.acquireTokenSilent({
            ...loginRegisterRequest,
            account: account
        }).then((value) => {
            setBearerToken(value.accessToken);
        }).catch((reason) => {
            toast({
                title: t("Login.Error.Title"),
                description: JSON.stringify(reason),
                status: "error",
                duration: 9000,
                isClosable: true,
                });
        });
    }

    return bearerToken;
};

export default useAccountBearerToken;
```

Joonis 15. Kasutaja sisselogimise *hook*-kood.

4.3.3 API päringute vigade kuvamine kasutajale

Interneti teel tehtud päringutel on mitu võimalikku põhjust ebaõnnestumiseks. Potentsiaalsetele rakendusepoolsetele vigadele lisaks võib tekkida ka rakenduse looja kontrolli alt väljas olevaid vigasid. Põhilised vead on näiteks nii kasutaja- kui ka teenusepoolsed riistvaralised vead, elektrikatkestused, seadmete konfiguratsiooni vead, turvavead, nagu DDoS rünnak, ja ka looduskatastroofid. Et selliste vigade puhul oleks kasutaja nendest teadlik, otsustas autor lisada päringute ebaõnnestumise kontrolli, mis kuvab kasutajale veateadet juhul, kui päring ebaõnnestub. Selles näites on esitatud React

Hooki kood, mis ümbritseb „useQuery“ React Hooki ning kuvab vea puhul kasutajale veateate toast elementi (Joonis 16).

```
import { useToast } from "@chakra-ui/react";
import { useTranslation } from "react-i18next";
import { QueryFunction, QueryKey, useQuery, UseQueryOptions, UseQueryResult }
 from "react-query";

const useQueryWithErrorHandler = <TQueryFnData extends unknown, TError extend
s unknown, TData extends TQueryFnData>(queryKey: QueryKey, queryFn: QueryFunc
tion<TQueryFnData>, options?: UseQueryOptions<TQueryFnData, TError, TData>):
UseQueryResult<TData, TError> => {
    const useQueryResult = useQuery(queryKey, queryFn, options);
    const { t } = useTranslation();
    const toast = useToast();

    if(useQueryResult.error && !useQueryResult.isLoading) {
        let message = t("Common.RequestError.Description.Unknown");

        if((useQueryResult.error as any).message) {
            message = (useQueryResult.error as any).message;
        }

        toast({
            title: t("Common.RequestError.Title"),
            description: message,
            status: "error",
            duration: 9000,
            isClosable: true,
            });
    }

    return {...useQueryResult}
};

export default useQueryWithErrorHandler;
```

Joonis 16. Päringu ümbritsemise veahaldusega kood.

4.3.4 Küsimuste ja vastuste pärimine andmebaasist

Kuna küsimused ja vastused on andmebaasis eraldi tabelitena, et neid oleks võimalikult mugav hallata, siis tuleb kasutajapoolsel API-l need eraldi pärida. Selleks, et vähendada nii ajalist kui ka rahalist kulu nende pärimisel, otsustas autor pärida kõik õppemooduliga seotud küsimused ühe päringuga, vastavate küsimustega seotud vastused teise päringuga ja siduda need serveris omavahel.

Küsimuste ja vastuste sidumine oleks kõige lihtsam kahekordse tsükli sees, kuid see tähendaks O(k*v) ajakeerukust. Et ajakeerukus küsimuste ja vastuste sidumisel oleks võimalikult väike, lõi autor räsitabeli, kus võti on küsimuse-Id ja väärtus on selle järgi rühmitatud vastused. See lahendus viis ajakeerukuse O(k+n) ajakeerukusele (Joonis 17).

```
var questions = await _questionRepository.TableNoTracking
    .Where(x => x.GroupId == learningModuleId)
    .ToListAsync();

var questionIds = questions.Select(x => x.Id).ToArray();

var answers = await _answerRepository.TableNoTracking
    .Where(x => questionIds.Contains(x.QuestionId))
    .ToListAsync();

await _entityTranslator.TranslateEntities(questions, languageId);
await _entityTranslator.TranslateEntities(answers, languageId);

var answersByQuestionId = answers
    .GroupBy(x => x.QuestionId)
    .ToDictionary(x => x.Key, x => x.ToArray());

return questions.Select(q => (Question: q, Answers: GetQuestionAnswers(q)))
    .ToList();

Answer[] GetQuestionAnswers(Question question)
{
    var questionAnswerOptions = answersByQuestionId[question.Id];

    var correctAnswers = questionAnswerOptions.Where(x => x.IsCorrectAnswer);
    var wrongAnswers = questionAnswerOptions.Where(x => !x.IsCorrectAnswer);

    return correctAnswers
        .Concat(wrongAnswers)
        .OrderBy(_ => Guid.NewGuid())
        .ToArray();
}
```

Joonis 17. Küsimuste ja vastuste andmebaasist pärimise kood.

4.4 Teenuste sidumine

4.4.1 Azure Blob storage

Rakendusel on kasutusel palju dünaamilisi faile. Esialgse MVP lahenduse puhul on võimalik lisada igale küsimusele pilt. Et kasutaja brauser saaks neid pilte võimalikult

kiirelt alla laadida ja neid faile eraldada CosmosDB andmebaasist, võttis autor kasutusele teenuse Azure Blob Storage. Selleks, et faile küsimuste lisamisel üles laadida, tegi autor teenuse, mis kasutab Azure Blob Storage'i-poolset lahendust BlobContainerClient. Selle initsialiseerimiseks on kasutatud seadistatud Azure Blob Storage'i „ConnectionString". Teenus laeb üles faili ja tagastab selle URL-i, mis läheb küsimuse külge ja mille järgi saab brauser otse pildi faile pärida (Joonis 18).

```
private readonly BlobContainerClient _blobContainerClient;

public BlobStorageService(string connectionString)
{
    var containerName = "content-images";

    _blobContainerClient = new BlobContainerClient(connectionString, containe
rName);
}

public async Task<string> UploadFileToBlobAsync(string fileName, Stream fileD
ata)
{
    var blob = _blobContainerClient.GetBlobClient(fileName);
    await blob.UploadAsync(fileData);

    return blob.Uri.AbsoluteUri;
}
```

Joonis 18. Azure Blob Storage'i faili üleslaadimise kood.

4.4.2 Azure AD B2C

Azure AD B2C kasutamiseks tuli esmalt luua Azure Portal keskkonnas Azure AD B2C Tenant ja lisada sellele eelistatavad seadistused. Selleks, et API päringuid oleks võimalik autentida, tuli lisada vastavad parameetrid ASP NET Core'i sisseehitatud autentimisele (Joonis 19).

```
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddMicrosoftIdentityWebApi(jwtOptions =>
    {
        Configuration.Bind("AzureAd", jwtOptions);

        jwtOptions.MetadataAddress =
            $"{Configuration["AzureAd:Instance"]}{Configuration["AzureAd:Doma
in"]}/B2C_1_react_sign_in_sign_up/v2.0/.well-known/openid-configuration";
    }, identityOptions => {
        Configuration.Bind("AzureAd", identityOptions);
    });
```

Joonis 19. Azure AD B2C serveris konfigureerimise kood.

Samad seadistused tuli lisada ka kasutajapoolsele rakendusele, mis nende põhjal saab kasutaja viia Azure AD B2C sisselogimise/registreerimise vaatesse. Peale sisselogimist suunab Azure AD B2C kasutaja tagasi kasutajapoolsele rakendusele koos Bearer Tokeniga, mida saab kasutada API päringute autentimiseks (Joonis 20 ja Joonis 21).

```
export const msalConfig = {
    auth: {
        clientId: applicationID,
        authority: signInAuthority,
        knownAuthorities: knownAuthorities,
        redirectUri: reactRedirectUri,
        postLogoutRedirectUri: reactRedirectUri
    }
};
```

Joonis 20. Azure AD B2C kliendipoolses rakenduses konfigureerimise kood.

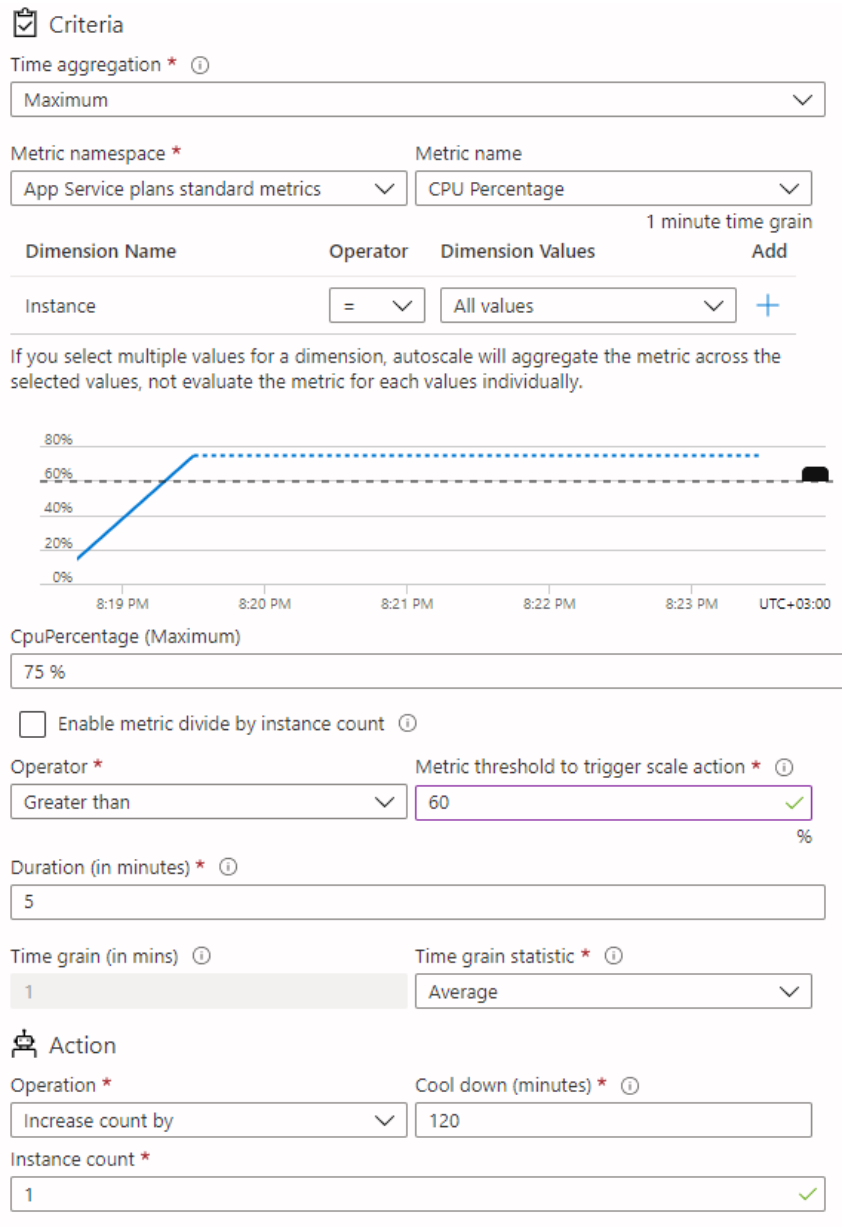
```
const msalInstance = new PublicClientApplication(msalConfig);

class App extends React.Component<Record<string, unknown>, undefined>
{
  public render() {
    return (
      <MsalProvider instance={msalInstance}>
        ...
      </MsalProvider>
    );
  }
}
```

Joonis 21. Azure AD B2C kliendipoolse konfiguratsiooni kasutamise kood.

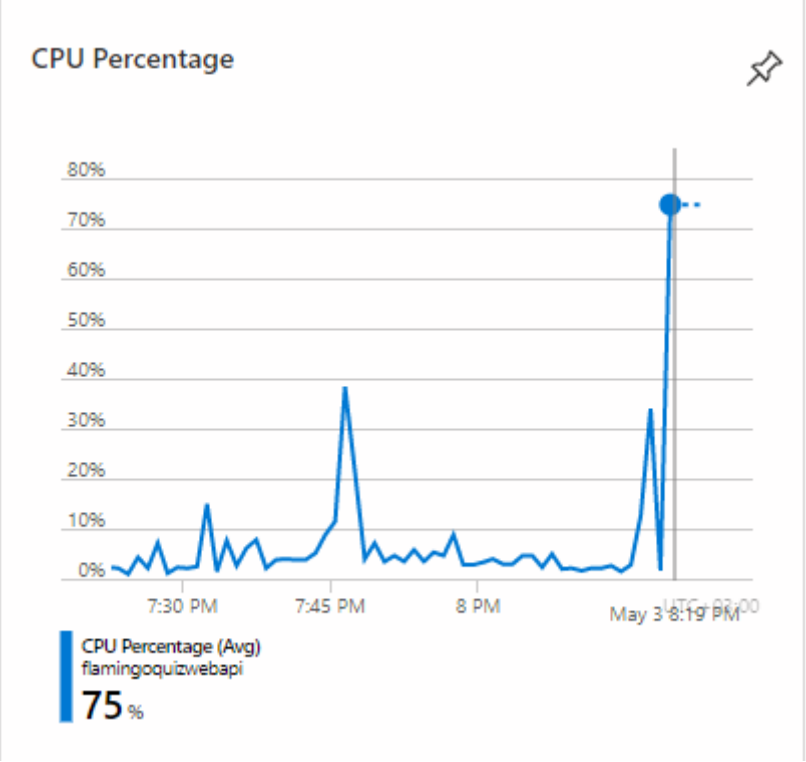
4.4.3 Automaatse skaleeruvuse seadistamine ja testimine

Kuna üks osa rakenduse nõudest on skaleeruvus, siis selles peatükis on ka kirjeldatud skaleruuvuse automatiseerimise seadistamine. Selleks seadis autor automaatse skaleeruvuse tingimuseks protsessori 60% kasutatavuse (Joonis 22). Ehk juhul kui rakendust majutaval instantsil suureneb protsessori kasutus üle 60%, lisab Azure automaatselt juurde uue instantsi, mis võtab päringuid vastu.



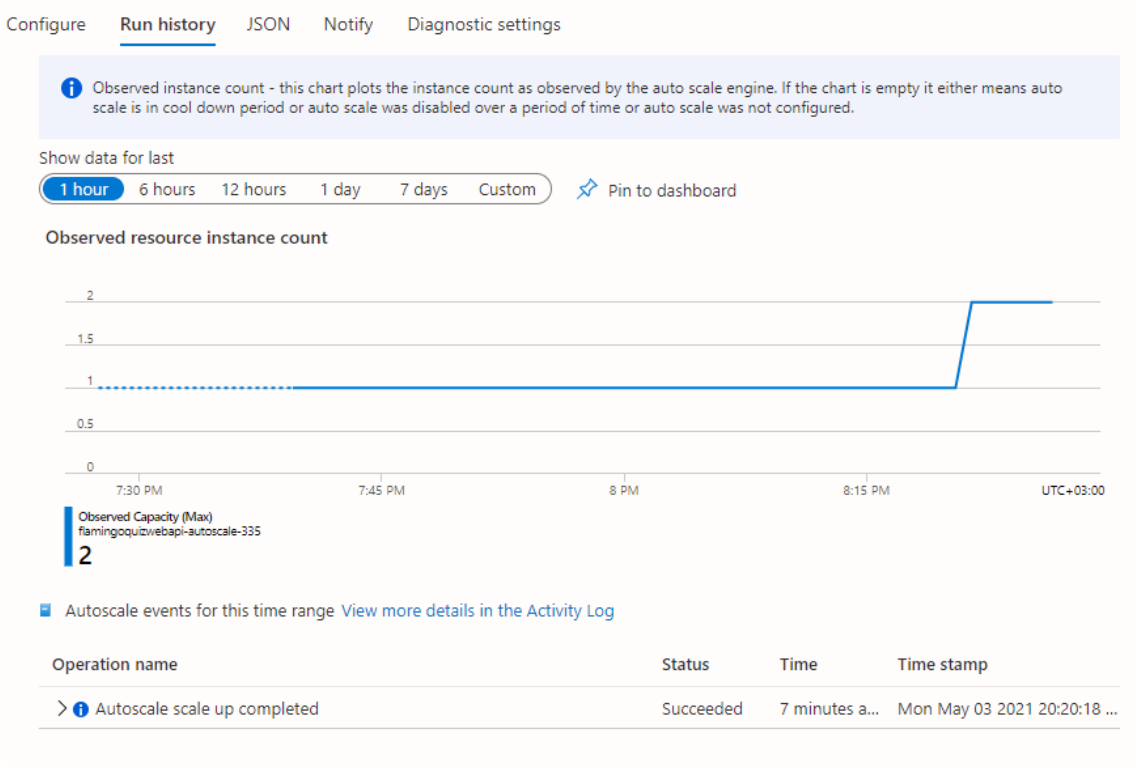
Joonis 22. Azure'i automaatse skaleeruvuse seadistamine.

Selleks, et katsetada seadistatud automaatset skaleerimist, kasutas autor teenust LoadView, mis võimaldab pilveteenuse kaudu saata suure koguse päringuid teenuse pihta, et katsetada selle käitumist suurema koormuse all (Joonis 23).



Joonis 23. Veebirakendust majutava instantsi protsessorikasutus koormuse rakendamisel.

Peale koormuse rakendamise jälgis autor Azure'i keskkonnas veebiteenuse instantside koguse muutuse graafikut, et kinnitada, kas automaatne skaleeruvus töötab. Graafikult on näha, et kui rakendusele lisati koormust, suurenes samas ajavahemikus automaatselt ka veebirakendust majutavate instantside arv 1 võrra (Joonis 24).



Joonis 24. Veebirakendust majutavate instantside arv ajas.

5 Edasiarendused ja parandused

Kuna lõputöö eesmärk oli luua vaid minimaalne töötav rakendus, siis selles peatükis on kirjeldatud olulisemaid edasiarendusi.

5.1 Mängulisuse aspektid

Rakenduse põhisisu on valikvastustega küsimused, mis koosnevad pildist ja tekstist. Et need oleks võimalikult huvitavad ja mitmekesised, siis on eesmärk võimaldada ka rikkalikuma sisu ehk videote, animatsioonide ja audioklippide lisamine. Selle võimaluse lisamine on tehnoloogilisest vaatepunktist üsna lihtne, keeruline on pigem taolise sisu tootmine.

Et rakendus oleks muu hulgas võimalikult kaasahaarav, on planeeritud lisada ka mitmesuguseid saavutuste/edusammude loetelusid ja efekte.

- Läbitud teemade, õppemoodulite ja küsimuste tasemed. Näiteks auhind 10 küsimuse taseme täitmise eest.
- Ilma ühegi vale vastuseta õppemooduli/küsimustiku täitmise eraldi välja toomine.

5.2 Monitooring ja kasutajate tagasiside

Selleks, et oleks ülevaade sisu kasutatavusest ja kasutajate eelistustest, tuleb lisada rakendusele ka asjakohane ülevaade monitooringu ja kasutajate tagasiside näol. Monitooringu eesmärk on näidata, kui palju mingit sisu kasutajad vaatavad. Peale selle annab monitooring võimaluse saada ülevaade rakenduse ressursside kasutusest ja ka rakenduses tekkinud vigadest, et vigu oleks võimalik tuvastada ja arendusressursside puudumisel vigade parandamist prioriseerida.

Mugav kasutajate tagasiside võimaldab saada ka paremat ülevaadet sellest, milline sisu kasutajatele meeldib ja milline mitte. Selles rakenduses planeeritud vabatahtlik kasutaja tagasiside on iga mooduli lõpus, kus kasutaja saab hinnata viie tärni süsteemis, kuidas talle õppemoodul meeldis, ning võib kirjutada ka täpsustava kommentaari. Samuti tuleb lisada ka kasutaja jaoks võimalikult mugav ja intuitiivne võimalus lisada tagasisidet rakenduse vigade kohta.

6 Kokkuvõte

Kokkuvõtlikult näitab see bakalaureusetöö, kuidas disainida ja arendada skaleeruvat veebipõhist õppeplatvormi kasutades pilvetehnoloogiaid. Bakalaureusetöö põhiline eesmärk oli luua veebipõhise õppeplatvormi minimaalne töötav rakendus, mis sai ka täidetud. Teema valimise põhjus seisnes mitmekesisema veebipõhise õppe meetodite vajaduses ja ka veebirakenduste skaleevuruse probleemides. Kogu tööprotsessi kirjeldus algas tausta selgitamisest, milles näidati taolise rakenduse vajadust, ja jätkus analüüsiga, kus autor võrdles sarnaseid rakendusi ning tõi välja kasutatavad tehnoloogiad ja arhitektuuri. Analüüsi järel kirjeldas autor praktilise osa kulgu.

Töö käigus loodud rakenduse tuum koosneb kolmest osast: teemad, teemade alla käivad õppemoodulid ja õppemoodulites olevad ülesanded. Kuna lõputöö eesmärk oli luua minimaalne töötav rakendus, siis ülesannete tüübina kasutati ainult küsimustikku. Praktilises osas sai ka vastavalt disainitud vaated ja nende põhjal juurutatud rakendus. Autor on rakenduse juurutamisest esitanud ka huvitavamad punktid.

Planeeritud rakendus oli edukalt analüüsitud, disainitud ja implementeeritud ning kõiki neid osasid on ka lõputöös kirjeldatud. Loodud rakendus on siiski vaid minimaalne versioon ning autoril on plaanis selle arendamisega ka edasi tegeleda.

Kasutatud kirjandus

[1] L. Bakken, N. Brown, ja B. Downing, „Early childhood education: The long-term benefits“, *J. Res. Child. Educ.*, kd 31, nr 2, 2017.

[2] L. Kirtman, „Online versus in-class courses: An examination of differences in learning outcomes.“, *Issues Teach. Educ.*, kd 18, nr 2, 2009.

[3] „5 Advantages And Disadvantages Of Online Learning“. https://www.easy-lms.com/knowledge-center/lms-knowledge-center/advantages-and-disadvantages-of-online-learning/item12529 (vaadatud apr 25, 2021).

[4] „Adverse consequences of school closures“. https://en.unesco.org/covid19/educationresponse/consequences (vaadatud apr 25, 2021).

[5] „Scaling Horizontally vs. Scaling Vertically“, *Section*, juuli 24, 2020. https://www.section.io/blog/scaling-horizontally-vs-vertically/ (vaadatud apr 25, 2021).

[6] „Cloud Computing Definition“. https://www.investopedia.com/terms/c/cloud-computing.asp (vaadatud mai 15, 2021).

[7] „Strong versus weak typing“, *Cornell University Department of Computer Science*. http://www.cs.cornell.edu/courses/cs1130/2011fa/module1/module1part4/strongtyping.html (vaadatud apr 25, 2021).

[8] Adam Freeman, *Pro ASP.NET Core MVC 2*, Seventh Edition.

[9] „React – A JavaScript library for building user interfaces“. https://reactjs.org/ (vaadatud apr 25, 2021).

[10] „Typed JavaScript at Any Scale.“ https://www.typescriptlang.org/ (vaadatud apr 25, 2021).

[11] „OpenAPI Specification - Version 3.0.3“. https://swagger.io/specification/ (vaadatud apr 25, 2021).

[12] Rico Suter, „NSwag: The Swagger/OpenAPI toolchain for .NET, ASP.NET Core and TypeScript“, *Github*. https://github.com/RicoSuter/NSwag (vaadatud apr 25, 2021).

[13] „What is Azure Active Directory B2C?“ https://docs.microsoft.com/en-us/azure/active-directory-b2c/overview (vaadatud apr 25, 2021).

[14] „OWASP Top Ten Web Application Security Risks | OWASP“. https://owasp.org/www-project-top-ten/ (vaadatud mai 04, 2021).

[15] „Introduction to Azure Cosmos DB“. https://docs.microsoft.com/en-us/azure/cosmos-db/introduction (vaadatud apr 25, 2021).

[16] „What Is Load Balancing? How Load Balancers Work“. https://www.nginx.com/resources/glossary/load-balancing/ (vaadatud apr 25, 2021).

[17] Martin Kleppmann, *Designing Data-Intensive Applications*, Twelfth release.

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

Mina, Ergo Posti

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Skaleeruv veebipõhine mänguline õppeplatvorm lastele“. mille juhendaja on German Mumma
   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.

02.05.2021

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.