TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Tõnis Bender T112159

ISESEISVA ÕPPE TÖÖVAHENDI LOOMINE VEEBIRAKENDUSENA

Bakalaureusetöö

Juhendaja: Kristina Murtazin
MSc

Tallinn 2017

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: Tõnis Bender

22.05.2017

Annotatsioon

Antud töö eesmärgiks on luua funktsionaalne veebirakenduse prototüüp mis võimaldab luua haridusliku eesmärgiga materjalide- ja küsimustekogumikke ning neid läbida kindla protsessi jooksul, mis sisaldab pidevat kordamist kõigi eelnevalt läbitud materjalide kohta küsimuste esitamise näol. Rakenduse andmesalvestuskihi lahenduse valik peaks tagamaks kiiret arendusprotsessi ning lahenduse itereerimist.

Peamine lahendatav probleem on luua eeldused kirjeldatud õppeprotsessi efektiivsuse põhjalikumaks testimiseks ning kohandamiseks kasutades optimaalseid ning jätkusuutlike tehnoloogiaid.

Töö tulemusena valminud veebirakendus võimaldab nõutud protsesse läbida, kuid väikse hulga potentsiaalsete kasutajate peal testimine näitas, et eelnevalt defineeritud nõuded rakendusele võivad tuntavalt muutuda esmaste kasutusmuljete tõttu. Saadud tõdemus kinnitas, et oli õige valik esialgu luua rakendus võimalikult lihtsa andmesalvestuskihiga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 36 leheküljel, 7 peatükki, 15 joonist, 1 tabelit.

Abstract

Creating a tool for self-studying as a web application

The goal of this thesis is to create a functional prototype of a web application that allows users to create lessons (sets of learning materials and questions) that can be shared to other users who can use the lesson for self-studying. The process of going through the lesson is built up in a way that enforces constant repetition by asking questions not only about the most recently passed material but also about previous materials. This goal serves the purpose of providing an acceptable and functional application which would allow to test the effectiveness of the learning process.

The main problem that the author attempts to solve is using optimal technologies and abstractions that help reach a testable solution quickly but also provide a good foundation to further develop the application.

The outcome of this thesis is a functional web application implementing the described learning process and material management with a simple data store based on a single JSON file. Modifications and fixes were made according to initial feedback from a few users and ideas for the next iterations of the application were considered.

The thesis is in Estonian and contains 36 pages of text, 7 chapters, 15 figures, and 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
CORS	*Cross-Origin Resource Sharing*, mehhanism, mis võimaldab veebilehel kasutada ressursse domeenidest mis erinevad veebilehe enda domeenist
DOM	*Document Object Model*, W3C poolt standardiseeritud HTML dokumentidega suhtlemise liides
ES6	*ECMAScript 6*, JavaScript keele spetsifikatsiooni kuues versioon, millega lisandus palju uut süntaksit
HTML	*HyperText Markup Language*, märgenduskeel mis defineerib veebilehe struktuuri
JAR	*Java Archive*, kokkupakitud fail mis sisaldab komplekti Java klassifailidest koos seonduvate metaandmete ning ressurssidega
JSON	*JavaScript Object Notation*, kergesti inimloetav andmevahetusformaat
JSX	JavaScript keele süntaksi täiendus, mis võimaldab kirjutada HTML sarnast struktuuri mille sees on võimalik kasutada JavaScripti ning mis töödeldakse React teegi funktsioonide väljakutseteks. Kasutatud React komponentide struktuuri kirjeldamiseks.
LMS	*Learning Management System*, tarkvaralahendus õppematerjalide ning õppeprotsesside haldamiseks
LRS	*Learning Record Store*, andmehoiusüsteem õpitegevuste sissekannete jaoks
REST	*Representational State Transfer*, Roy Fielding poolt defineeritud veebiteenuste arhitektuur [1]
W3C	*World Wide Web Consortium*, veebistandardeid välja töötav organisatsioon
Õpiobjekt	Digitaalne interaktiivne õppematerjal, mis on taaskasutatav, terviklik, toetab õppimist ning vastab tehnilisetele standarditele [2]

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Taust ja probleem

Iseseisvaks õppimiseks on antud töö kirjutamise hetkel palju rohkem võimalusi kui oli neid mõned aastakümned tagasi. Interneti kaudu saab ligi üüratule hulgale infole näiteks Vikipeedia, Google Scholar või lihtsalt otsingumootori abil. Selliste vahenditega saab küll leida vajalikke ja huvi pakkuvaid materjale, kuid mõne teema sügavamalt läbi töötamiseks ja omandamiseks oleks abiks konkreetse teabe komplekteerimine ning interaktiivseks tegemine enesekontrolli lihtsustamiseks. Keskkonnad nagu Moodle, Coursera, Khan Academy, edX pakuvad võimalust veebis kursuseid läbida sarnaselt ülikoolikursustega, kuid need võivad osutuda liiga mahukaks õpilase jaoks või liiga raskesti rakendatavaks õppematerjali loojale. Lihtsama alternatiivina leidub ka rakendusi nagu Easygenerator, mille abil saab luua läbitavaid õppematerjale, mis võivad koosneda paljudest erinevatest sisutüüpidest ning teadmiste kontrollidest. Selliseid lahendusi on küll palju, kuid materjalide koostamisele ning läbimisele leidub veel uudseid lähenemisi, mida tasub proovida.

1.2 Ülesande püstitus

Töö raames on eesmärgiks realiseerida veebirakendus, mis võimaldab koostada õppematerjale ja küsimusi nende kohta ning seejärel komplekteeritud tulemust jagada teistele inimestele läbimiseks. Läbimise protsessi peab välja töötama komplektis esinevate materjalide pideva kordamise põhimõttel vastavalt sellele kuidas õpilasel saadud teadmised kinnistunud on.

1.3 Metoodika

Püstitatud eesmärke proovib töö autor saavutada samm-sammulise protsessi läbi. Esiteks on kavas analüüsida ning kirja panna süsteemi baasfunktsionaalsuse. Seejärel saab tuletada peamised rakenduse kuvad, esialgse andmemudeli ning kliendi ja serveri vahelise liidese. Funktsionaalsuse realiseerimiseks ning testimiseks valib autor erinevate

raamistike ja tehnoloogiate hulgast kasutatavad tööriistad tagamaks esiteks kiiret lahenduseni jõudmist ning silmas pidades tehtud valikute jätkusuutlikust juhul kui loodud projekt kasvab suuremaks. Saadud tulemust testib autor iseseisvalt ning ka küsib tagasisidet mõnelt rakenduse potentsiaalselt kasutajalt.

1.4 Ülevaade tööst

Käesoleva töö sisu on jaotatud kuueks osaks. Esimene ehk antud peatükk selgitab töö tausta, ülesande püstitust ning selle saavutamise metoodikat ning pakub ülevaadet töö peatükkidest. Teises peatükis kirjeldatakse esialgseid nõudeid süsteemi funktsionaalsusele ning võrreldakse sarnaste olemasolevate lahendustega. Kolmandas peatükis on uurimise all kasutajaliidese programmeerimiseks kasutatava üldise raamistiku valimine. Neljas peatükk sisaldab kaalutlusi rakenduse andmetalletuskihi simuleerimiseks või realiseerimiseks antud töö jaoks. Viies peatükk käsitleb konkreetseid lahendusi ning lisandunud tehnoloogiate kasutuskohti funktsionaalse prototüübi valmimisel. Kuuendas ehk viimases peatükis hinnatakse loodud rakendust. Kirjeldatakse ka kasutajatelt saadud tagasisidet ning parandusi ja muudatusi mis rakendusse sisse viidud said.

2 Ülevaade loodavast süsteemist ja alternatiividest

Käesolevas jaotises kirjeldab töö autor loodava süsteemi võimalusi ning protsesse. Võrdluseks on toodud ka sarnaseid lahendusi.

2.1 Nõuded süsteemile

Rakenduse sihtgrupiks on valdavalt kooliõpetajad ning õpilased. Õpetajad kui materjalide loojad ning komplekteerijad peavad saama süsteemis teha järgmisi toiminguid:

- Sisse ning välja logida
- Luua ning muuta materjalikomplekte (edaspidi nimetatud kui tund), mis koosnevad sisuosadest, küsimustest ja vastusevariantidest. Need komplektid peaksid vastama üldjoontes õpiobjekti omadustele [2]
- Jagada loodud tunde lingi kaudu teistele kasutajatele läbimiseks
- Kontrollida kes ja millal on tunni läbinud

Tunni läbimine toimub järgnevalt:

- Tunnis sisalduvaid materjale näidatakse kasutajale ükshaaval ette vastavalt tunni loomisel määratud järjekorrale. Pilt- ja tekstmaterjali puhul on kasutajal võimalik liikuda küsimuste faasi, video puhul peab vaatama klipi lõpuni.
- Küsimuste faasis näidatakse kasutajale üht suvalist valikvastustega küsimust äsja läbitud sisu kohta. Lisaks näidatakse kaht suvalist küsimust kõigi eelnevalt läbitud materjalide kohta.
- Vea tegemisel viiakse kasutaja tagasi materjali vaatamise faasi vastavalt küsimusele mille puhul eksiti. Tehes mitmes küsimuses vea, läbitakse kõik vastavad materjalid enne kui minnakse järgmise juurde. Eksitud materjali kohta

näidatakse üht suvalist küsimust ehk mitte tingimata seda sama mille puhul eksiti.

Kirjeldatud protsessi eesmärgiks on pideva kordamise ning juhuslike küsimuste küsimise kaudu aidata kaasa materjali kinnistumisele.

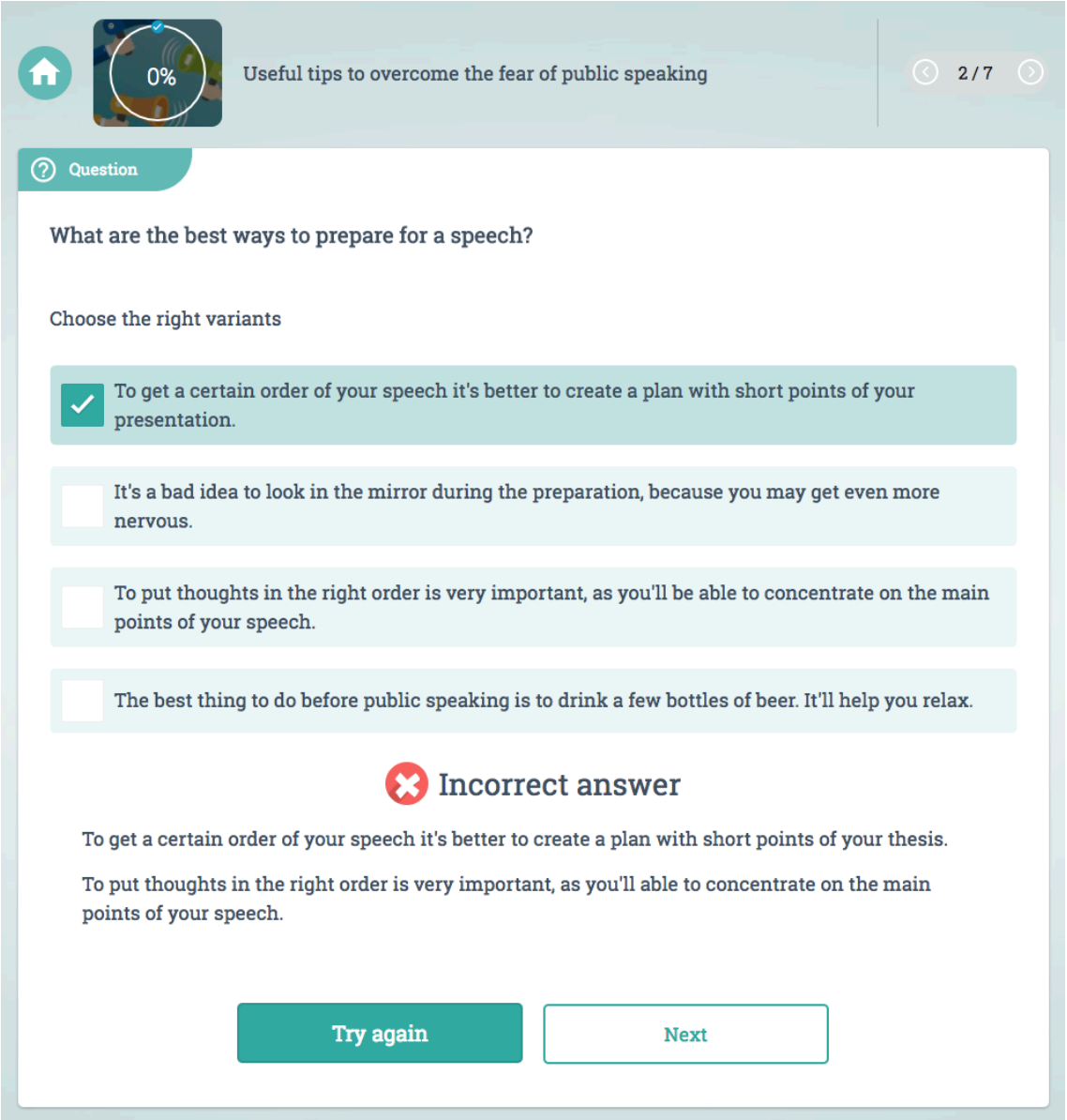
2.2 Võrdlus sarnaste süsteemidega

Antud peatükis uurib autor milliseid võimalusi pakuvad teised tarkvaralahendused õppematerjalide koostamiseks ning läbimiseks. Põhirõhk on läbimise funktsionaalsusel, kuna see on kesksel kohal antud töö tulemusena valminud prototüübi testimisel.

2.2.1 Easygenerator

Easygenerator on veebirakendus mis võimaldab koostada õppematerjale paljude erinevate sisutüüpide ning teadmistekontrollidega. Viimastest võib näidetena tuua üksiku ning mitme valikvastusega küsimuse, lünkade täitmise, seoste kokku viimise ning lause tõeväärtuse määramise. Loodud komplekte saab järgnevatel viisidel eksportida: lingiga süsteemi pakutavasse pilveteenusesse, iframe elemendina mida saab olemasolevasse veebilehte sisestada, standardse paketina mida saab LMS-ides kasutada ning oma serveris tarnitava failide komplektina.

Materjalide läbimise protsess on Easygenerator puhul üles ehitatud nii, et kasutaja saab vabalt navigatsiooniribal liikuda komplekti erinevate elementide vahel, olgu selleks kas sisu- või küsimuste osa. Vastamise ebaõnnestumise korral saab kohe tagasiside ning võimaluse uuesti proovida (Joonis 1). Komplekt arvestatakse läbituks kõigi küsimustesektsioonide edukal läbimisel. Kogutulemust ning individuaalseid vastuseid on võimalik saata Easygenerator süsteemi või ise defineeritud LRS-i.



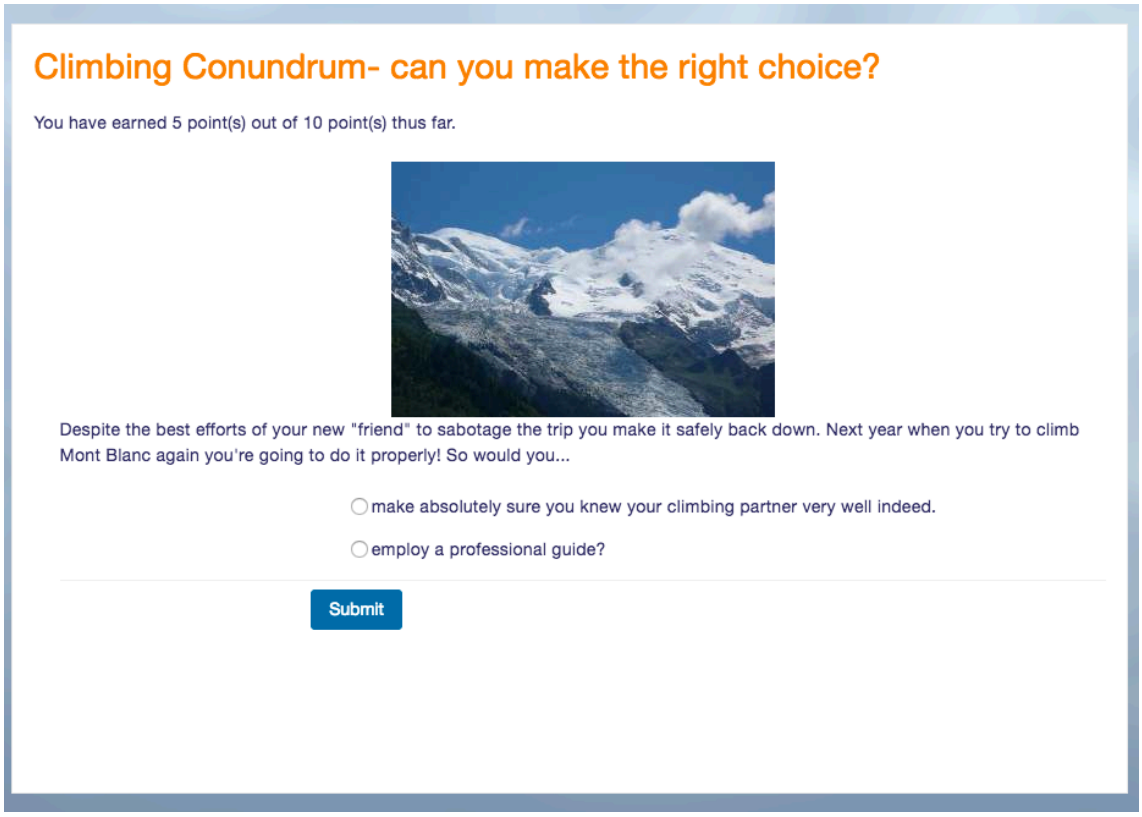
Joonis 1. Easygenerator süsteemis küsimusele vastamine.

Easygenerator ei võimalda teistsuguseid materjalide läbimise viise nagu antud töö tulemusena valmivas prototüübis, kus materjalid läbitakse järjestatult ning küsitakse pidevalt kordavaid küsimusi ka eelnevate osade kohta.

2.2.2 Moodle *lesson* funktsionaalsus

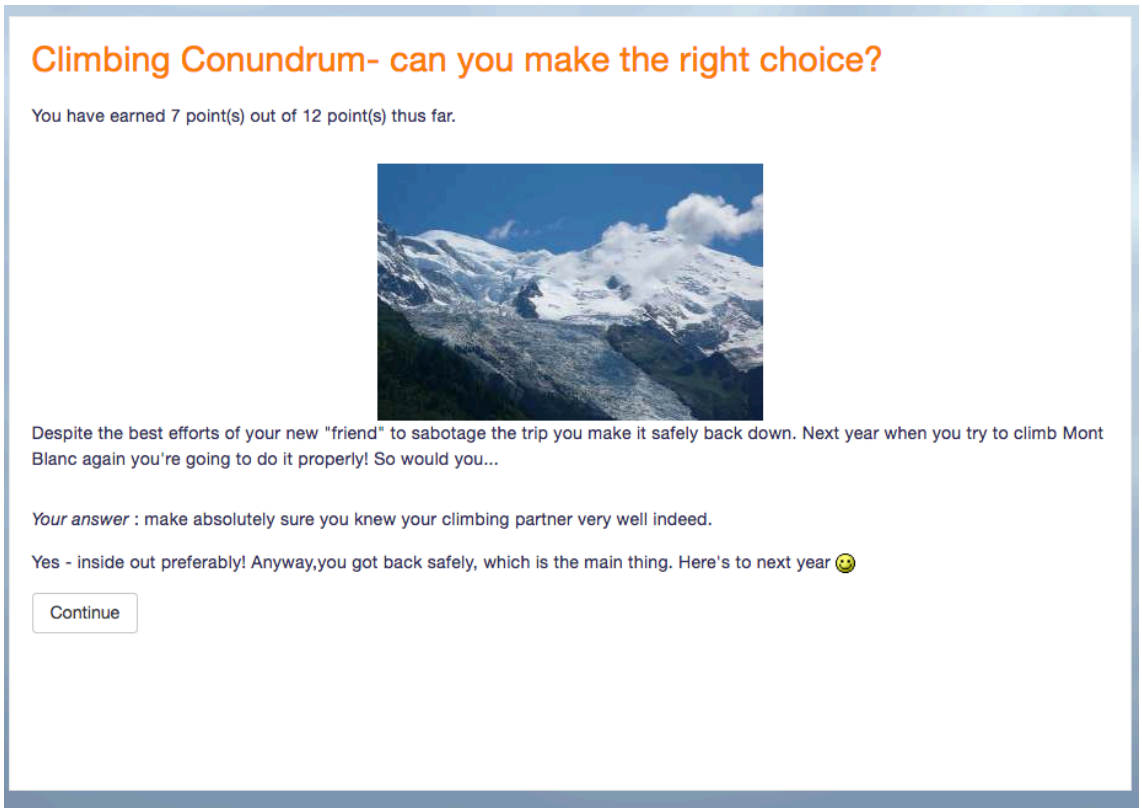
Moodle on laialt levinud modulaarse lähenemisega LMS ehk materjalide ning õppeprotsesside haldamise süsteem. Selle kasutajad saavad luua kursuseid mis võivad sisaldada erinevaid materjale ning tegevusi. Kursusele registreerinud kasutajad saavad sisu kasutada vastaval kursuse looja seadistustele.

Üheks võimalikuks sisuelemendiks Moodle kursusel on tund ehk *lesson* [3]. See on moodul, mis esitleb läbijale seeria HTML lehtedest millel on sisu- ning küsimusteala (Joonis 2).



Joonis 2. Moodle *lesson* ehk tunni sammu läbimine.

Vastamisel saab tagasisidet, mida tunni loojal on võimalik seadistada ning seejärel saab kasutaja suunata järgmisesse sammu (Joonis 3) või vea tegemisel pakkuda uut võimalust vastamiseks. Looja saab ka jälgida tunni läbimise ning individuaalsete soorituste ning küsimuste statistikat.



Joonis 3. Moodle *lesson* ehk tunni sammu tulemus.

Kirjeldatud lahendus on küll osaliselt sarnane antud töö raames realiseeritavale prototüübile, kuid see ei võimalda läbitud sammude materjalide kohta kordavaid küsimusi esitada ning nendes vigade tegemisel tagasi materjali juurde minna. Lisaks on selle funktsionaalsuse haldamine üsna keeruline ning eeldab Moodle süsteemi kasutamist.

3 Ülevaade kasutajaliidese tehnoloogiatest

Järgnevalt kirjeldab töö autor vahendeid mis prototüübi ehitamiseks said valitud. Otsuste tegemisel lähtuti kaalutud projektide näilisest elujõulisusest ning isiklikust huvist valitud teemadega lähemalt tutvuda.

3.1 React

React on Facebooki poolt arendatud deklaratiivne ning komponendipõhine JavaScripti teek kasutajaliideste ehitamiseks. Tänapäeval on enamik populaarseimast JavaScripti raamistikest võtnud komponendipõhise lähenemise, kuid React oli üks esimestest lahendustest mis populariseeris seda ning on ka jätkuvalt väga menukas – näiteks hiljutisel StackOverflow poolt läbi viidud tarkvarainimeste küsitlustel saavutas React kõige armastatumate raamistike, teekide ja muude tehnoloogiate kategoorias esimese koha, kus 66,9% vastanutest märkis, et soovib selle tehnoloogiaga ka edaspidi tegeleda [4].

3.1.1 React põhimõtted

Komponent on React puhul defineeritud kui iseseisev ning eraldatud tükk kasutajaliidesest mida saab isolatsioonis käsitleda [5]. Tehniliselt on see JavaScript klass, mida kasutaja loodud komponendid laiendavad ning implementeerivad minimaalselt *render* meetodi. Lisaks on võimalik komponendil kasutada erinevaid elutsükli meetodeid mida kutsutakse kas komponendi lisamisel DOM-i, selle uuendamisel või DOM-ist eemaldamisel. Komponendil võib olla sisemine olek ehk *state* ja sisendatribuudid ehk *props*, mille muutumistele oskab React reageerida ning uuesti *render* meetodit välja kutsuda. Tulemuseks on komponentide ahel, kus andmed liiguvad ühesuunaliselt ning rakenduse olekut on võimalik hõlpsamalt mõista ning järgida kui näiteks kahesuunalise andmete liikumise puhul, kus lapskomponent võib vanemalt saadud andmeid muuta.

Efektiivselt DOM-i muudatuste tegemiseks kasutab React meetodit, mille puhul salvestatakse nii öelda virtuaalne DOM JavaScripti objektina mällu ning komponentide oleku või atribuutide muutumisel võrreldakse olemasolevat ning uut mälupilti DOM-ist. Nende erinemisel muudetakse ka kasutajale nähtavat DOM-i. Sellise lähenemisega

hoiab React võimalike DOM elementide muutuste ning pärimiste arvu võimalikult madalal, kuna nendega tegelemine on rohkem ressursse nõudvam tegevus kui JavaScript objektide võrdlemine.

Kasutajaliidese struktuuri kirjeldamiseks on soovitatud kuid ei pea kasutama JavaScripti keele täiendust nimega JSX. See on HTML-i struktuuriga sarnanev vahend, mis kompileerub React teegi JavaScript funktsioonide väljakutseteks mis omakorda loovad päris HTML struktuuri. Selline lahendus nõuab küll eraldi õppimist, kuid lubab komponendi struktuuri kirja panemisel kasutada JavaScripti võimalusi näiteks nimekirjade üle itereerimisel või valikulisel kuvamisel, aidates kaasa programmeerija produktiivsusele.

3.1.2 React Native kasutamise potentsiaal

React Native võimaldab luua natiivseid iOS ja Android mobiilirakendusi kasutades JavaScripti ja Reacti. See defineerib React komponendid mis tõlgenduvad vastava platvormi ehitusblokkideks, võimaldades sama koodibaasiga ehitada rakendusi erinevatele seadmetele. Käesolevas töös loodud React veebirakendust ei saa küll otseselt React Native rakenduse loomisel kasutada, kuid kindlasti on hõlpsam React Native rakendust ehitada kui funktsionaalsust peegeldav veebirakendus kasutab sarnaseid tehnoloogiaid. Selle kaalutluse tõttu on ka Reacti valimine prototüübi jaoks kindlam valik autori seisukohalt.

3.1.3 Alternatiivid

Valitud kasutajaliidese realiseerimise tehnoloogiale oli alternatiividena kaalumise all kaks raamistikku:

- Vue on võrdlemisi uus nähtus populaarsete JavaScript raamistike hulgas, kuid kogub kiiresti hoogu kasvus. Vue pakub sarnaselt React-iga komponendipõhist lähenemist, kuid kasutab kuvade struktureerimiseks tavalist HTML-i mis võib sisaldada raamistikuspetsiifilisi osi nagu näiteks atribuute elementidel. Veel on sarnased aspektid virtuaalse DOM-i kasutamine ning keskendumine tuumfunktsionaalsusele, jättes spetsiifilisemad mured nagu globaalne olekuhaldus ning URL põhise navigeerimise teiste teekide hooleks [6]. Kokkuvõttes on Vue väga huvitav tehnoloogia ja oli tõsine kandidaat valikus, kuid jäi React-ile alla küpsuse kaalutlusel.

- Angular on Google poolt loodud JavaScript raamistiku teine versioon. See on esimesest versioonist (nüüdseks kutsutud kui AngularJS) märgatavalt erinev. Erinevalt eelnevast versioonist mille puhul kasutati palju kahesuunalist andmete sidumist võeti Angulari teises versioonis kasutusele sarnane lähenemine nagu React puhul – ühesuunaline andmete sidumine ja komponendipõhine lähenemine. Lisaks on selle arenduses väga sügavalt juurdunud TypeScript keele kasutamine, mis transpileerub JavaScriptiks ning võimaldab lisada tüübikontrolle ja muud JavaScripti koodile. Sarnaselt esimese versiooniga on Angular tuntavalt suurema API-ga ja seega on ka keerulisem kui React või Vue. Antud projekti raames eelistas autor React teegi modulaarsemat ja lihtsamat lähenemist.

4 Serveri tehnoloogiavalik

Veebirakendustes on serveriga ühendumiseks üldjuhul kasutusel REST teenused ning sama on kavas ka antud töö raames rakendada. Teine kaalutud variant oli GraphQL, mis on uudsem lähenemine Facebooki poolt ning mis lubab serveriliidese vastu kasutada päringukeelt. Sellisel viisil ei pea arendama eraldi ressursse ega otspunkte ning saab anda kliendile paindlikuma ligipääsu andmetele. Kuna see lahendus on vähem küps ning mitte laialt kasutatud nagu REST liides, siis jäi viimane siiski valituks.

4.1.1 Staatilised andmed

Üks võimalus prototüübi kiireks arenduseks on kirja panna kasutatavate REST ressursside näidisvastused projekti failistruktuuri vastavalt disainitud liidesele ning arenduskeskkonnas jooksutades edastada päringud liidesele nendele failidele. Kirjeldatud meetodit saab rakendada näiteks serve-static vahevaraga NodeJS peal põhineval Express serveris. Viimast saab testkeskkonnas eraldiseisvalt jooksutada ning vahevara saab ka arenduskeskkonnas hõlpsalt kasutusele võtta. See lahendus aga ei võimalda andmeid salvestada ning seetõttu langes valiku hulgast välja, kuna eesmärgiks on funktsionaalne prototüüp kus kasutaja saaks ka oma koostatud tunde läbida.

4.1.2 Relatsiooniline andmebaas ja REST liides sellele

Teine variant on luua andmebaas ning seda kasutav back-end rakendus näiteks Java veebiraamistikku Spring Boot kasutades. Spring Boot abil on võimalik oma back-end rakendus ühte käivitatavasse JAR-i komplekteerida, mis sisaldab ka rakendusserverit nagu näiteks Tomcat ning on seadistatud võimalikult palju vastavalt levinud tavadele, eemaldades arendajalt palju tööd.

Selle lähenemise positiivne külg on kogu süsteemi terviklik arenemine. Samas lisab andmebaasi ning sellega ühenduva rakenduse loomine ja seadistamine märgatavalt keerukust ning töötava kasutajaliideseni jõudmine võtab selle võrra rohkem aega.

4.1.3 JSON Server

JSON server on Express serverina jooksev rakendus, mis kasutab JSON faili andmebaasina ning pakub REST liidest selle faili sisu pärimiseks ja muutmiseks [7].

Andmefailiga liidestumiseks on kasutusel sama autori poolt loodud Lowdb, mis kasutab JavaScripti utiliitide teeki Lodash päringute võimaldamiseks JSON andmestikule.

Joonis 4 kujutab näitena toodud andmefaili ja Tabel 1 sees on loetletud mõned päringud mida andmestikule on võimalik teha.

```
{
  "lessons": [
    {
      "id": 1,
      "name": "Keskaeg",
      "description": "Seitsmenda klassi ajaloomaterjal"
    }
  ]
}
```

Joonis 4. JSON Server andmefaili näide

Tabel 1. Võimalikud päringud JSON Server REST liidesele

Päringu meetod	Päring	Päringu keha	Tulemus
GET	/lessons/1	-	Massiivi lessons kirje millel on id=1
GET	/lessons?name=Keskaeg	-	Nimekiri kõigist lessons massiivi elementidest, millel on name=Keskaeg
POST	/lessons	`{ "name": "Uusaeg", "description": "Test" }`	Lisatakse päringu keha kirje lessons massiivi. ID väli genereeritakse automaatsel JSON server poolt
PATCH	/lessons/1	`{ "name": "Keskaeg 2" }`	Massiivi lessons kirjel millel on id=1 muudetakse name väli vastavalt päringu kehale
DELETE	/lessons/1	-	Massiivist eemaldatakse kirje millel on id=1

Selline lahendus võimaldab lihtsa vaevaga luua REST konventsioonidele vastavad ressursid ning neid vajadusel arendusprotsessis kergesti kohandada. Lahenduse paindlikkuse ning pakutava arenduskiiruse tõttu osutus see ka valituks funktsionaalse prototüübi arenduseks.

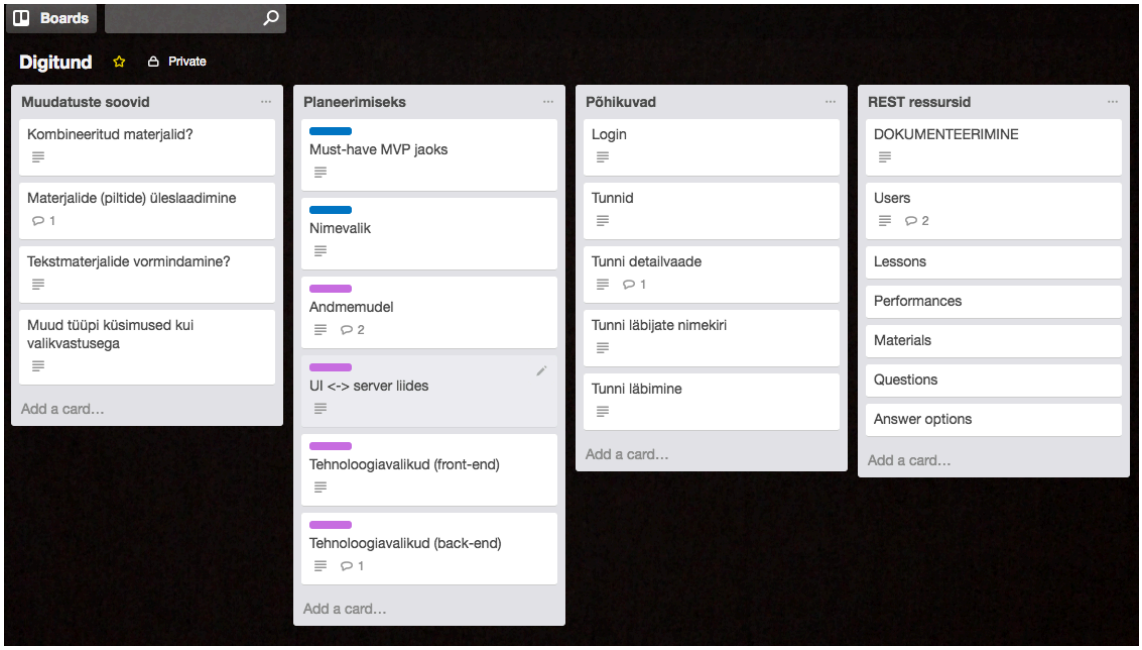
5 Funktsionaalse prototüübi arendus

5.1 Kasutatud tööriistad

Käesolevas peatükis loetleb autor projekti arendamiseks kasutatavaid vahendeid ning kirjeldab nende rolli ning vajalikkust.

5.1.1 Trello

Esialgsete mõtete koondamiseks ja nende viimistlemiseks kasutas autor projektihaldus tarkvara Trello. See on kanban metoodikast inspireeritud rakendus, mille puhul projekt on kujutatud tahvlina mis jaguneb listideks ning mis omakorda koosnevad kaartidest. Levinud kasutusviisiks on horisontaalselt kõrvuti asetsevad listid järjestada vasakult paremale vastavalt ülesannete staatusele arendusprotsessis nagu näiteks ‘Ootel’, ‘Analüüsis’, ‘Arenduses’, ‘Testimises’, ‘Kinnitatud’, ‘Tarnitud’. Trello ei sea mingeid piiranguid kasutamise viisile, vaid pakub võimalust kasutada ennast täpselt nii nagu kasutaja heaks arvab – näiteks esitleda suuremaid teemasid eraldi listidena (Joonis 5).



Joonis 5. Trello tahvel prototüübi planeerimiseks.

Antud töö puhul kasutas autor Trellot ülevaate saamiseks rakenduse nõuetele, põhikuvadele ning REST liidestele. Lisaks sai ka testimise jaoks tehtud kasutajatele tagasiside jaoks eraldi list kuhu mõtteid grupeerida.

5.1.2 Visual Studio Code

Visual Studio Code on Microsofti poolt 2016. aastal ametlikult avalikustatud koodiredaktor, mis põhineb Electron tehnoloogial. See osutus valituks suure valiku laienduste, integreeritud terminali ja versioonihalduse toe ning kasutajaskonna hea tagasiside tõttu.

5.2 Kasutajaliides

5.2.1 Projekti ülesseadmine

React projektide loomise lihtsustamiseks on Facebooki poolt loodud käsurealt käivitatav vahend create-react-app, mille abil saab luua esialgse projekti struktuuri koos soovitatud sõltuvuste ning konfiguratsiooniga. See lahendus võtab oma vastutuse alla suure osa seadistamisest mis tavaliselt React projekti loomisel tuleks arendajal ise selgeks teha ja paika sättida [8], näiteks:

- React, JSX ja ES6 süntaksite tugi Babel abil
- Mõndade JavaScripti keele lisade tugi mida ES6 spetsifikatsioonis veel ei ole
- Arendusserver mis võimaldab aktiivset koodi jälgimist ning koodi staatilist analüüsi, andes arendajale koheselt hoiatusi ning veateateid nii serveri väljundlogis kui ka brauseris
- Projekti moodulite kokkupakkimine webpack toel, millel on näiteks järgmised hüved [9]:
    - Võimaldab JavaScripti failides importida staatilisi ressursse nagu pildid või CSS failid
    - Optimeerib projekti väljundfailide suurust erinevate strateegiatega, millest üks on ainult projekti lähtekoodis imporditud moodulite kaasamine väljundisse
    - Väga kiire inkrementaalne kompileerimine aitab arendusprotsessi kiirena hoida

- CSS reeglitele automaatne prefikside lisamine ühtlase stiliseerimise saavutamiseks mitmetes brauserites
- Valmis skriptid projekti jooksutamiseks, testide käivitamiseks ning tarnitava versiooni ehitamiseks

5.2.2 Versioonihaldus

Koodi versioonihalduseks on kasutusel Git. See on üks levinumaid versioonihaldussüsteeme ning seetõttu sai ka valituks antud projekti raames. Repositooriumi internetis kättesaadavaks tegemiseks on see üles laetud GitLab veebipõhisesse keskkonda, mida saab kasutada ka projekti dokumenteerimiseks ning probleemide halduseks.

5.2.3 Rakenduse olekuhaldus

Rakenduste kasutamise jooksul on üldjuhul vaja säilitada infot hetkeoleku kohta ning peab ka olema võimalus seda olekut hallata. Näiteks peab telefonivälja komponent võimaldama kasutajal sisestada oma telefoninumbri ning seda meeles pidama. React komponentides on võimalik seda realiseerida kas telefonivälja komponendi sisese olekuna või komponendile edastatava atribuudina. Esimese puhul on komponendi jaoks telefonivälja väärtus muudetav (Joonis 6) kuid teise puhul peab komponent käsitlema sisendina saadud atribuuti kui mittemuudetavana ning saab ainult reageerida selle väärtuse muutumisele ning peab teavitama atribuudi muutmise üritamise sündmusest vanemkomponenti (Joonis 7).

```
import React, { Component } from 'react';

class PhoneField extends Component {
  state = { phoneNumber }

  handleChange = event => {
    this.setState({ phoneNumber: event.target.value });
  }

  render() {
    return (
      <div>
        <label for="phoneNumber">Sisesta telefoninumber:</label>
        <input
          id="phoneNumber"
          value={this.state.phoneNumber}
          onChange={this.handleChange} />
      </div>
    );
  }
}

export default PhoneField;
```

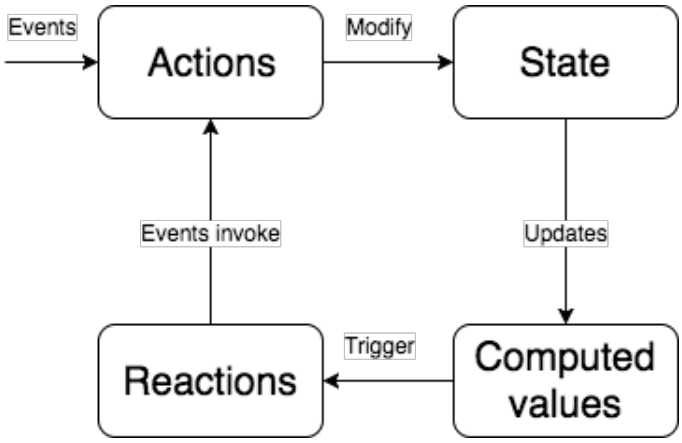
Joonis 6. Komponendisisese olekuhaldusega React komponent.

```
import React from 'react';
import PropTypes from 'prop-types';

const PhoneField = (props) => (
  <div>
    <label for="phoneNumber">Sisesta telefoninumber:</label>
    <input
      id="phoneNumber"
      value={props.phoneNumber}
      onChange={props.onPhoneNumberChange} />
  </div>
)

PhoneField.propTypes = {
  phoneNumber: PropTypes.string.isRequired,
  onPhoneNumberChange: PropTypes.func.isRequired
}

export default PhoneField;
```

Joonis 7. Olekuhaldus komponendis mis ei oma sisest olekut.

Keerulisema olekuga rakenduse puhul võib eelnevalt kirjeldatud React pakutav olekuhaldus jääda liiga piiravaks ning raskesti hallatavaks, kuna sügava komponentide ahela puhul kõige vanemast komponendist oleku viimine ahela viimase komponendini

nõuab selle oleku edastamist igas lülis ning muudatuste tegemine selles lülis võib olla tülikas. Selle mure lahendamiseks jaoks on antud projektis kasutusele võetud MobX teek, mille abil saab viia andmed tsentraalsesse hoidlasse. Komponente saab panna reageerima MobX hoidlas jälgitavatele andmetele vastava märgistusega ning komponendid saavad välja kutsuda hoidlas defineeritud toiminguid (*action*) andmete muutmiseks või muude kõrvalmõjude saavutamiseks. Hoidlas saab veel defineerida arvutuslikke väärtusi mis kasutavad jälgitavaid andmeid, et neid mingil teisel kujul tagastada. Näiteks võib tuua jälgitava massiivi filtreerimine. Selliste arvutuslike väärtuste puhul on oluline, et arvutusfunktsioon oleks puhas ehk ilma kõrvalmõjudeta [9].



Joonis 8. MobX olekuhalduse diagramm.

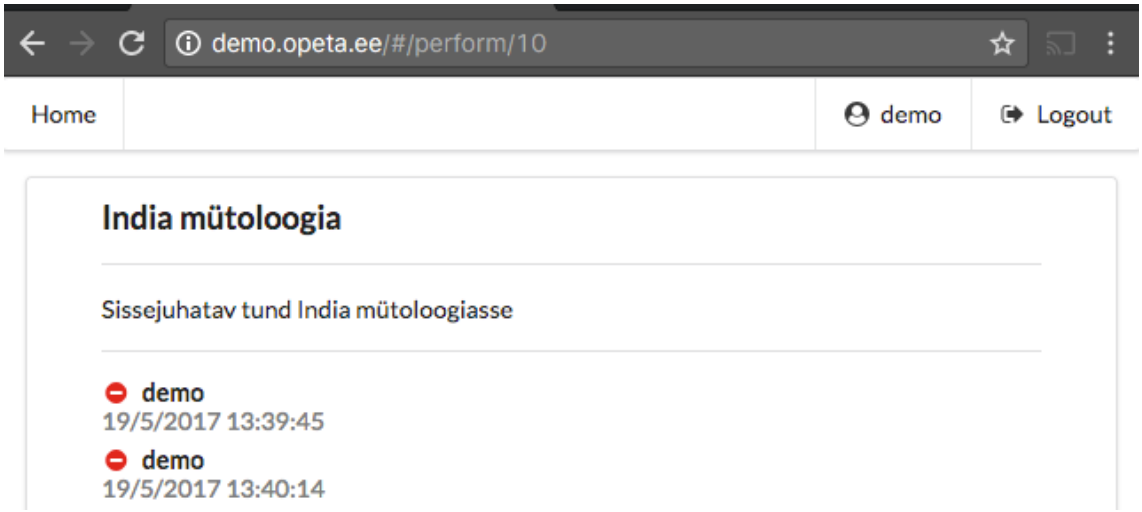
MobX kasutamiseks on kaks võimalust:

- Märgistada MobX hoidlad ja jälgijad dekoraatoritega, mis on uudne JavaScripti süntaks (ES.Next ehk järgmise ECMAScript spetsifikatsiooni kandidaat) ning seetõttu vajab lisaseadistust projektis kasutamiseks. Käesolevas töös tuli kasutusele võtta modifitseeritud versioon create-react-app baasfunktsionaalsust pakkuvatest skriptidest [10]. Kuigi seadistuste kohandamine oli lisavaev ja dekoraatorite süntaks pole veel kinnitatud standardis, siis leidis autor et kasutusmugavusest tulev võit on piisav dekoraatorite rakendamiseks.

- Jälgitavate väljade ja jälgivate komponentide ümbritsemine MobX funktsiooni väljakutsega mis annab analoogse tulemuse dekoraatoritega, kuid muudab märgistamise kohmakamaks kuna peab jälgima, et ümbritsevad sulud saaksid õigesti pandud.

5.2.4 URL põhine navigeerimine

Kuigi loodav prototüüp on üheleherakendus ehk SPA, on siiski vaja, et kindlatele kuvadele oleks võimalik URL-i kaudu navigeerida. Selle jaoks peab olema mehhanism, mis oskaks jälgida brauseris sisestatud URL-i ning kuvaks vastavalt defineeritud komponenti rakenduses. Käesolevas töös on see vajadus eriti aktuaalne tundide jagamise puhul, kus tunni koostaja jagatud lingi kaudu peab jõudma tunni läbimise kuvale (Joonis 9).



Joonis 9. URL põhine navigeerimine prototüübis.

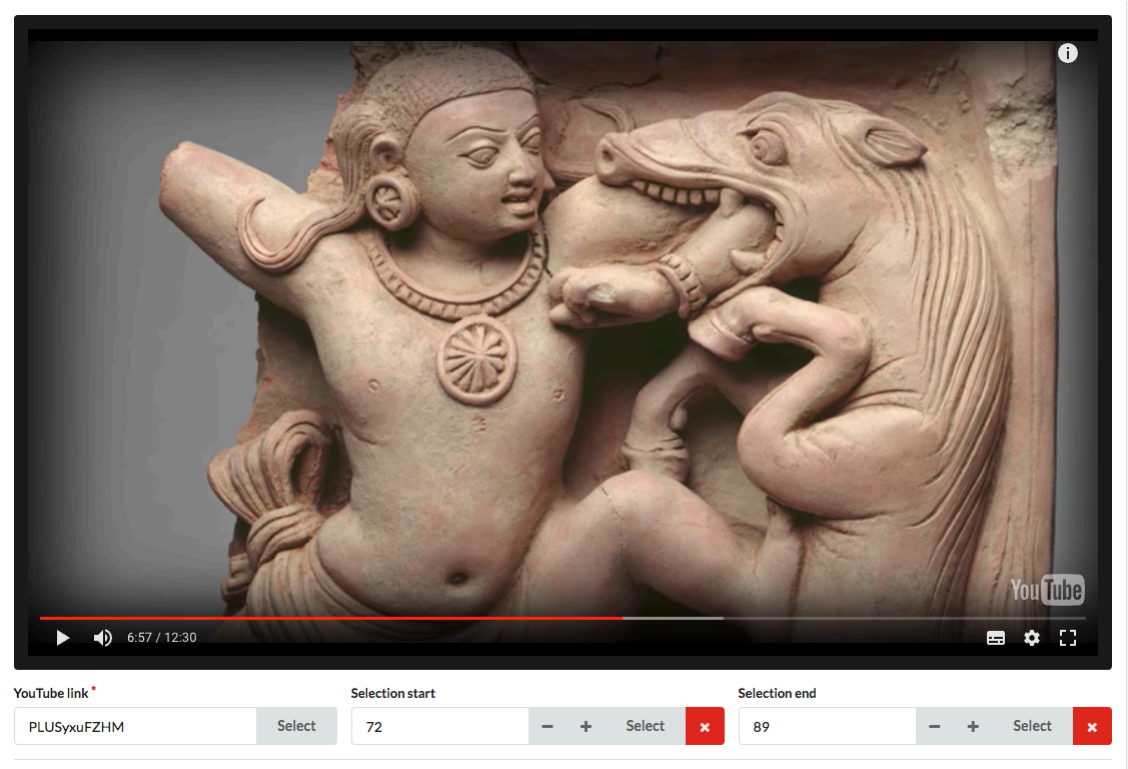
Kirjeldatud navigeerimist võimaldab antud töös React Router v4. Joonis 9 näitab kuidas kasutusel on selline variant ruuterist, mis kasutab # märki ning sellest järgnevat URL-i osa kuva valimiseks. Teine võimalus on ruuter, mis teeb sama kuid ilma # märgita. See nõuab rakendust pakkuva serveri seadistamist selliselt, et nähes päringut http://demo.opeta.ee/perform/10 ei prooviks server pärijale pakkuda faili /perform/10 vaid suunataks päring /index.html lehele, kus jookseb rakendus mis oskab selle URL-i põhjal õiget kuva näidata [11].

5.2.5 Videomaterjalide lisamine ja läbimine

Antud töös on suur tähtsus videomaterjalide haldamisele ja läbimisele sobiva lahenduse leidmine.

Haldamise puhul on tähtis võimalikult lihtsalt võimaldada video lisamist materjali sisse ning videost mingi lõigu välja valimist. Viimane nõue on tähtis, kuna pidevalt kordamisküsimusi esitav protsess eeldab üsna lühikesi videojuppe, kuid õppematerjalina loodud video on arvatavasti tervikuna liiga pikk. Valituks sai YouTube videotele

viitamise lubamine, kuna see on kõige rohkem kasutatud platvorm selleks otstarbeks ning on tasuta teenus. Kuigi suvalisele veebilehele sisse poogitav YouTube mängija ei ole väga hõlpsalt seadistatav vastavalt rakenduse erinõudmistele, õnnestus siiski luua vajalik liides videomaterjalide loomiseks (Joonis 10). Liides võimaldab valida video kas URL-i või YouTube sisese ID põhjal ning määrata klipi lõpu- ning algusaega vastavalt video hetkeajale või sekundi kaupa edasi või tagasi nihutades.



Joonis 10. Videomaterjali haldusliides.

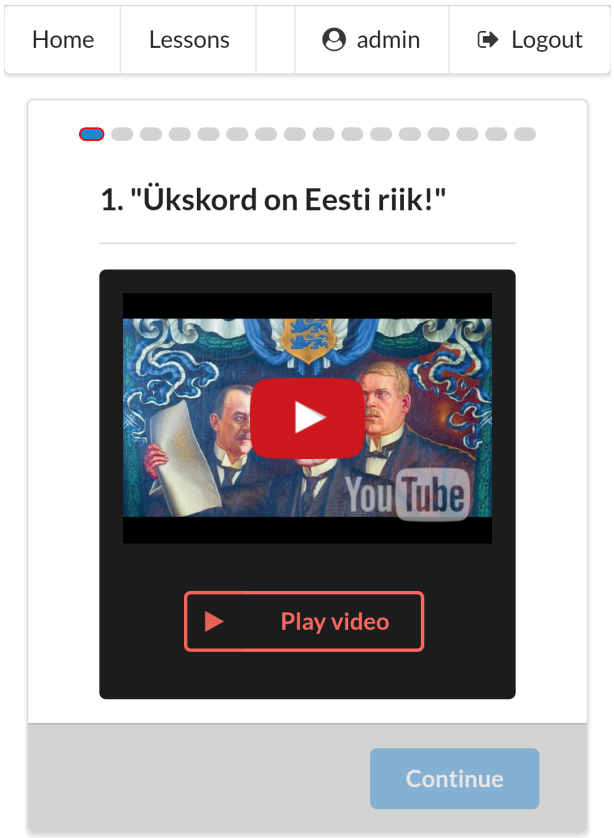
Videomaterjali läbimise liidese jaoks lahenduse leidmine oli keerulisem. Rakendusele püstitatud nõuete kohaselt ei tohi kasutaja video esitust manipuleerida, et vältida kasutaja kiiret edasi kerimist ja vastamisel huupi pakkumist. Selle jaoks sai video elemendile määratud CSS-i reegel:

```
.performance-video {
  pointer-events: none;
}
```

Joonis 11. Video esitamisel kasutaja interaktsiooni keelamiseks kasutatav CSS reegel.

Selline lahendus ei ole ideaalne, sest teadlik kasutaja võib brauseris selle reegli eemaldada, kuid antud prototüübi jaoks sobis esialgu kirjeldatud lähenemine. Lisaks on veel mängijale edastatud parameetrid video automaatselt mängimiseks ja algus- ning lõpuaja määramiseks. Veel on mängija seadistatud visuaalselt võimalikult

minimalistlikuks [12]. Mobiilsete seadmete tarbeks oli veel vaja lisada mängija väline nupp (Joonis 12) millega saaks video käivitada, kuna üldjuhul ei toimi mobiilsetes seadmetes automaatse mängimise funktsionaalsus.



Joonis 12. Videomaterjali läbimise liides mobiilis.

Kasutatud meetoditega õnnestus tagada videomaterjalide haldamise ning nende läbimise funktsionaalsus, kuid tulevikus on arvatavasti vaja toetada ka videote üleslaadimist ning rakendada mängijale suuremaid kohandusi. Sellisel juhul on võimalus kasutada teenust nagu Wistia mis on küll tasuline kuid pakub suuremat paindlikkust [14].

5.2.6 UI komponendid

Tänapäeval on oluline, et veebirakendused oleksid funktsionaalsed ning näeksid head välja ka mobiilsetel seadmetel. Lisaks oleks lahenduse kiirel prototüüpimisel hea, kui ei peaks liiga palju vaeva nägema visuaalse poole sättimisel, vaid saaks kasutada mõistliku vaikimisi välimusega komponente. Mõlemale murele annab hea esialgse lahenduse mõne UI raamistiku kasutusele võtmine. Populaarseimad valikud nende hulgast on näiteks Bootstrap, Foundation ja Semantic UI. Käesoleva prototüübi jaoks valis autor Semantic UI, kuna sellel on raamistiku loojate poolt ametlikult tunnustatud versioon

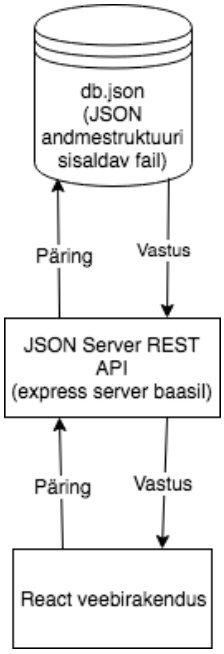
saadaval mis integreerub React teegiga. See versioon ei kasuta interaktsioonide jaoks laialt levinud JavaScript teeki jQuery nagu teeb seda Semantic UI ise, vaid implementeerib vastava funktsionaalsuse silmas pidades Reacti virtuaalset DOM-i, sest jQuery kasutamine tähendab lehe reaalse DOM-i muutmist mis omakorda viib virtuaalse ning reaalse DOM-i sünkroonist välja [15]. jQuery eemaldamise tõttu ei ole Semantic UI React versioonis kõik funktsionaalsus saadaval, kuid antud projekti jaoks ei ole see probleem.

Semantic UI on vaikimisi erinevatele ekraanimõõtudele kohanev ning pakub laia valiku erinevaid komponente mida kasutajaliidese ehitamisel saab ära kasutada. Üks põhilistest komponentidest mida UI raamistikud pakuvad on võre ehk *grid*, mis aitab lehte jagada kindla laiusega osadeks. Seda pakub ka Semantic UI, jagades lehe 16 osaks ning võimaldades määrata komponentidele laiust vahemikus 1 kuni 16.

5.3 Server

5.3.1 Arhitektuur

Rakenduse arhitektuur on väga lihtne kuna tegemist on prototüübiga.

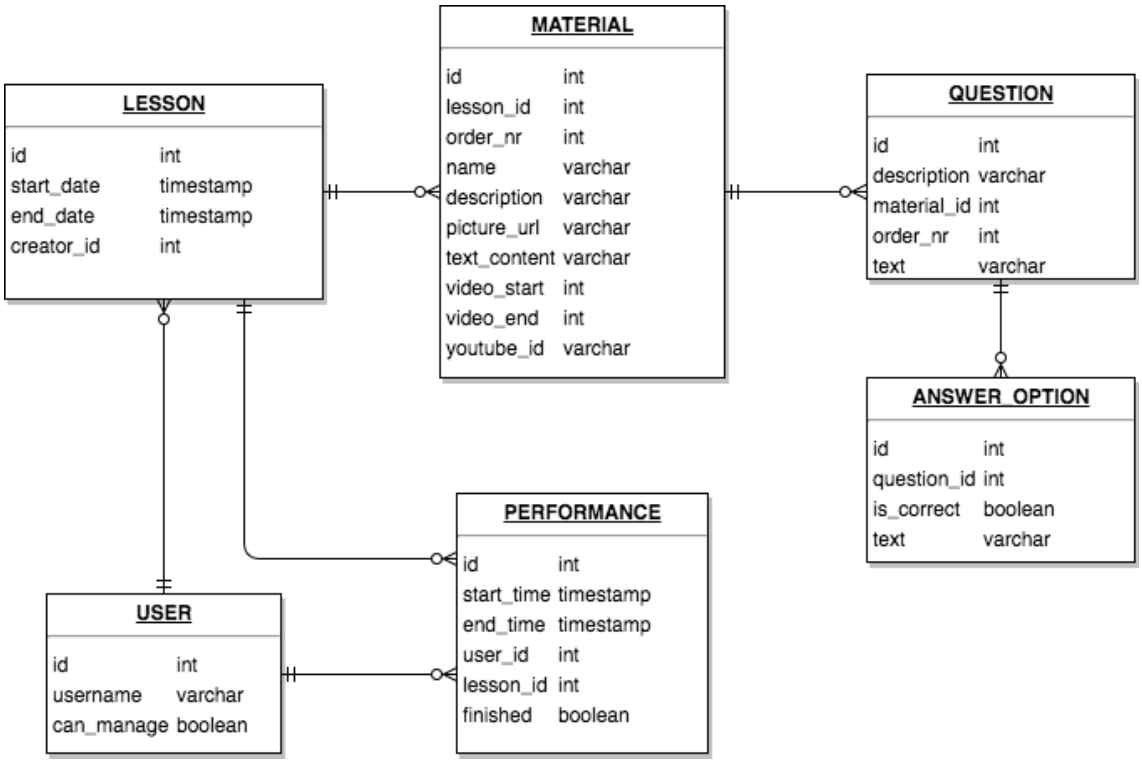


Joonis 13. Prototüübi arhitektuur.

Arhitektuur koosneb ainult kolmest osast: andmeid talletav JSON fail, seda haldav JSON Server rakendus mis pakub REST liidest ning viimast kasutav veebirakendus ehk klient (Joonis 13). Protsesside ning andmemudeli kindlamaks muutumisel on võimalus vahetada JSON Server ja selle poolt kasutatav andmefail suvalise andmebaasi ning sellega liidestuva rakenduse vastu välja. Selleks juhuks on autor arvestanud Spring Boot Java rakenduse ning PostgreSQL andmebaasi kasutamisega, kuid see jääb antud töö skoobist välja.

5.3.2 Andmemudel

Kuna antud prototüübi valmistamise raames ei valmi andmebaas, siis sai kirja pandud ainult ligikaudne andmemudel vajalike olemite ja nende sisuga (**Error! Reference source not found.**).



Joonis 14. Esialgne rakenduse andmemudel.

Puudub autentimise ja autoriseerimise võimalus ning olemid on kirjeldatud sellisena nagu neid kasutajaliidese pool vaja läheb. Sellegi poolest on mudel kasulik esialgse kontseptsiooni loomiseks mida saab päris back-endi realiseerimisel kohandada.

5.3.3 REST liides JSON Server abil

JSON Server REST liides pakub kõigile defineeritud olemitele standardseid REST ressursside meetodeid nagu GET, POST, PUT, DELETE, PATCH. JSON Server on vaikimisi seadistatud vältima CORS probleeme, lubades läbi teisest domeenist tulevaid kasutajaliidese päringuid. Serverit saab ka koos teiste Express serveri moodulitega koos jooksutada ning on võimalik päringuid ning vastuseid töödelda. Näiteks on prototüübi arendusel kasutatud võimalust lisada POST meetodiga objektide loomisel külge loomise aeg (Joonis 15).

```
const jsonServer = require('json-server')
const server = jsonServer.create()
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()
server.use(middlewares)

// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)
server.use((req, res, next) => {
  if (req.method === 'POST') {
    req.body.createdAt = Date.now()
  }
  // Continue to JSON Server router
  next()
})

server.use(router)
server.listen(3001, () => {
  console.log('JSON Server is running')
})
```

Joonis 15. JSON Serveri kohandatud käivitusskript loodud objektidele loomise aja lisamiseks.

Arendusprotsessis kasutatava veebirakendust pakkuva serveri ning JSON serveri kõrvuti käivitamine on realiseeritud create-react-app käivitusskriptide kohandamise teel. Selle jaoks on kasutusel JavaScripti teek concurrently mis võimaldab ühe terminalikäsuga jooksutada mitut protsessi. Lisaks on create-react-app seadistuses määratud *proxy* mis suunab serveripäringud kasutajaliidesest JSON serverisse [15]. Selline lahendus tagab väga lihtsa liidestuse serveriga nii arenduses kui ka tarneserveris.

6 Tulemuse testimine

Käesolevas peatükis vaatleb autor funktsionaalse kasutajaliidese testimist võimalike lõppkasutajatega ning tagasisidest tulenevate muudatuste sisu.

6.1 Piiratud kasutajagrupi tagasiside

Tundide haldamise osa õnnestus testida ühe põhikooli ajalooõpetaja peal ning läbimise osa veel paari teise isiku peal.

6.1.1 Tundide haldamine

Testimise käigus tuvastasid testijad mõned vead rakenduses – teatud juhul uute materjalide loomisel salvestati topeltkirjeid. Arenduses tehtud testimine ei paljastanud viga, sest protsessi läbiti väga sarnasel viisil mis tegelikult lõppkasutaja käitumisega ei olnud piisavalt vastavuses.

Funktsionaalsuse osas sai realiseeritud lisasoovid video haldamise jaoks. Sooviks oli täpsem kontroll video alguse ning lõpu valimisel ning lahenduseks lisati video algus- ning lõpuaja sekundi võrra edasi ning tagasi nihutamise nupud. Veel täiendati YouTube lingi sisestusvälja parsimisfunktsionaalsusega mis lubab sisestada YouTube URL-e nii lühidal kui ka pikal kujul ning samas töötab ka ainult YouTube video ID puhul. Kohandati ka tunni läbijate nimekirja niimoodi, et näidataks ainult esimest edukat tunni läbimist iga kasutaja kohta. Realiseerimata jäi soov piltmaterjale üles laadida oma arvutist ning võimalus komplekteerida mitut erinevat materjalitüüpi kokku. Viimane soov kulmineerus ideeks, mis on rakenduse järgmise iteratsiooni suuremaks sisuks – nimelt materjalide ning küsimuste samal tasemel kohtlemine, mis tähendaks küsimuste lahtisidumist konkreetse materjali küljest ning materjalide läbimisel kohelda küsimusi kui eelneva üksiku materjali või mitme materjali kohta käivaks. Antud juhul on sellise soovi realiseerimine selle võrra lihtsam, et pole veel konkreetset andmebaasi ning back-end lahendust mida tuleks kohandama hakata.

6.1.2 Tundide läbimine

Tundide läbimise protsessis esines ka vigu õige materjali juurde naasmises kui eksida küsimuses ning tunni läbimise progressiriba värvimises vastava materjali staatuse järgi. Need olid ka spetsiifilisemad juhud mille otsa arenduse käigus ei sattunud ning mis said

kähku parandatud. Testijad tuvastasid ka mobiilsete seadmetele spetsiifilise käitumise, mille puhul ei toiminud videomaterjalide automaatne käivitamine vastava parameetri lisamisel YouTube mängijale. Ei aidanud ka programmaatiliselt mängija käivitamine komponendi initsialiseerimisel ning lahenduseks sai mängijast eraldiseisva käivitamise nupu lisamine. Autor eemaldas ka võimaluse enne videomaterjali lõppemist jätkata küsimuste vooruga, kuna see pakkuvat liiga lihtsat viisi tunnist läbi kiirustada küsimustele suvaliselt vastates. Järgmisteks iteratsioonideks sai üles märgitud soov teistsuguste küsimustetüüpide lisamiseks nagu näiteks paaride ühendamine, rühmitamine ning lünktekst.

7 Kokkuvõte

Käesoleva töö sihiks oli luua funktsioneeriv ning lihtne õppevahendi prototüüp ühe kindla õppeprotsessi toetamiseks, mis keskendub materjalide inkrementaalsele läbimisele ning pidevale kontrollile ning kordamisele. Lisaks pidi süsteem võimaldama ka läbitavaid materjalidekogumikke ning küsimusi luua. Selle jaoks oli vaja ka võimalikult lihtsat andmete säilitamise lahendust, mis tagaks kõigi protsesside testitavuse.

Loodud React teegil põhinev veebirakendus realiseerib nii tunni looja kui ka läbija kasutusjuhud. Tunni looja saab määrata tunnile nime, kirjelduse ning avatud olemise ajavahemiku. Saab ka lisada tekst-, pilt- ja videomaterjale ning nende järjekorda tunnis ümber liigutada. Võimalik on materjalidele valikvastustega küsimuste lisamine ning nende valikvastuste haldamine. Tunni looja saab tunde jagada lingi kaudu teistele kasutajatele. Tunni läbija näeb avalehel oma varasemaid katseid ning saab tunde läbida vastavalt koostaja määratud järjekorrale. Iga materjali järel küsitakse käesoleva materjali kohta ühe küsimuse ning lisaks kaks küsimust suvaliselt eelnevate materjalide kohta. Vea tegemisel viiakse vastava materjali juurde ning parandamisel naasetakse esialgse vea tegemise positsioonilt. Kõigis protsessides andmete salvestamine tagati JSON andmefailil põhineva lihtsa serveri abil mille nimi on JSON Server.

Testimisel osalenud ajalooõpetaja kinnitusel võib esialgset lahendust sõnastatud tingimustele vastavaks nimetada. Sellegi poolest sai kinnitust ka teadmine, et süsteem vajab põhjalikku edasiarendust ning mõningast ümberkorraldust, et vastata kasutaja ootustele ning tulevikus võimaldada ka teistsuguseid õppeprotsesse läbida. Lõputöö raames jäi tegemata põhjalikum õppeprotsessi efektiivsuse testimine, mis tuleb kindlasti lahendusega jätkamisele eelnevalt läbi viia. Sellele järgnevalt on võimalik põhjalikumalt andmebaasile ning sellega liidestuvale rakendusele rõhku panna.

Kasutatud kirjandus

[1] R Fielding. Representational State Transfer (REST). [WWW]. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (22.05.2017)

[2] L. Pilt, A. Villems, T. Marandi E. Kampus. Õpiobjekt ja selle omadused. [WWW]. https://sisu.ut.ee/opiobjekt/1-mis-%C3%B5piobjekt (22.05.2017)

[3] Moodle. Lesson activity. [WWW]. https://docs.moodle.org/33/en/Lesson_activity (22.05.2017)

[4] Stack Overflow. Developer Survey Results 2017. [WWW]. https://insights.stackoverflow.com/survey/2017#most-popular-technologies (22.05.2017)

[5] Facebook. React Component. [WWW]. https://facebook.github.io/react/docs/react-component.html (22.05.2017)

[6] Vue.js. Comparison with Other Frameworks. [WWW]. https://vuejs.org/v2/guide/comparison.html (22.05.2017)

[7] Typicode. JSON Server. [WWW]. https://github.com/typicode/json-server#routes (22.05.2017)

[8] Facebook. create-react-app. [WWW]. https://github.com/facebookincubator/create-react-app (22.05.2017)

[9] Webpack. webpack. [WWW]. http://webpack.github.io/docs/ (22.05.2017)

[10] MobX. Computed decorator. [WWW]. https://mobx.js.org/refguide/computed-decorator.html (22.05.2017)

[11] K. Ristovski. Medium. [WWW]. https://medium.com/@kitze/configure-create-react-app-without-ejecting-d8450e96196a (22.05.2017)

[12] Facebook. Serving Apps with Client-Side Routing. [WWW]. https://github.com/facebookincubator/create-react-app/blob/master/packages/react-scripts/template/README.md#serving-apps-with-client-side-routing (22.05.2017)

[13] Google. YouTube Embedded Players and Player Parameters. [WWW]. https://developers.google.com/youtube/player_parameters (22.05.2017)

[14] Wistia. Wistia Player. [WWW]. https://wistia.com/product/player (22.05.2017)

[15] Semantic UI React. Introduction. [WWW]. https://react.semantic-ui.com/introduction (22.05.2017)

[16] A. Accomazzo. Using create-react-app with a server. [WWW]. https://www.fullstackreact.com/articles/using-create-react-app-with-a-server/ (22.05.2017)