TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mark Genrich Geller 185882IADB

ABIKURSUSEID PAKKUV VEEBIRAKENDUS

Bakalaurusetöö

Juhendaja: Jaanus Pöial
PhD

Kaasjuhendaja: Stefano Fiorenza
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: Mark Genrich Geller

16.04.2021

Annotatsioon

Käesoleva bakalaurusetöö peamine eesmärk on luua veebirakendus, mis käsitleb endas erinevaid abi- ja lisakursuseid. Rakendus on mõeldud nii kooliõpilastele, tudengitele, kui ka vanematele generatsioonidele ja inimestele, kes soovivad õppida, midagi uut juurde või leida endale hobi.

Veebirakenduse serveripoolne kood on kirjutatud kasutades Java programmeerimiskeelt ning selle Spring nimelist raamistiku. Kliendipoolne kood on valminud kasutades React.js raamistiku.

Arenguprotsessi käigus valminud rakenduses on olemas kahte tüüpi kasutajaid. Ühed, kes on niinimetatud õpetajate rollis, jagavad enda teadmisi ning postitavad lehele kursuseid, samaaegselt reklaamides nii ennast kui ka enda teadmisi. Teine kasutajate tüüp on aga õpilaste rollis, kes võivad valida endale vastavalt soovile sobiva abikursuse, mida sooviksid saada selgemaks või koguni algusest õppida. Lehel on võimalik kursuste sorteerimine, filtreerimine ning otsimine, mis teeb meelepärase valiku leidmise kergemaks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 49 leheküljel, 3 peatükki, 14 joonist, 2 tabelit.

Abstract

Web Application to Provide Variety of Courses

The primary goal of this thesis was to develop a web application which provides online courses for the users. It could benefit firstly, all the students with helping to learn all the difficult subjects from school. In addition to students, it would benefit also from extra knowledge gained from the courses provided in the application. The second group of people, who would benefit from this program, would be all those, who has something to share with the world, some of their knowledge, they would be the ones who would create new courses for the people on this website.

Application would be built using REST API architecture, which means that website would use client-server model. It would have separate front-end, which would be accessible by user and through which user could see all the courses. There would also be a possibility to filter them by the desired tag, which could limit the results. Also, user would be able to sort courses by rating, price or duration and even search by their title. Through front-end user could also be able to create new courses, leave a review under each completed course and even send feedback regarding the website to developers to let them now what is good and what could be better. Back-end on the other hand would have all the logic and functionality of the website and would be communicating with the database. Server side would be built using different layers like repository layer and service layer to ease the maintenance of the code and make it easier to avoid mistakes. It would be also beneficial to avoid making mistakes and would be recommended from the security point of view.

The thesis is in Estonian and contains 49 pages of text, 3 chapters, 14 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides.
CSS	*Cascading Style Sheets*, kaskaadilaadistik, mida kasutatakse veebilehe kujundamisel.
DTO	*Data transfer object*, lihtsustatud objekt andmete kandmiseks
GUID	*Globally Unique Identifier*, universaalne unikaalne identifikaator.
HTML	*HyperText Markup Language*, hüperteksti märgistuskeel.
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll, protokoll informatsiooni edastamiseks arvutivõrkude vahel.
JSON	*Javascript Object Notation*, lihtsustatud andmevahetusvorming.
REST	*Representational state transfer*, tarkvaraarhitektuuri laad.
UUID	*Universally Unique Identifier*, universaalne unikaalne identifikaator.

Sisukord

Jooniste loetelu

Tabelite loetelu

Sissejuhatus

Tänavu on tudengite ja õpilaste üheks põhiprobleemiks tööle saamine. Tööandjatele tihtipeale ei piisa enam vaid haridusastmest, mis on muutunud üsna tavaliseks inimeste seas, vaid tahetakse näha inimeste teadmisi ja kogemust antud alal. Tööle võetakse peamisel kandidaate, kellel on suurem oskuste pagas, mida ette näidata olgu see kas otseselt või kaudselt seotud tööga. Vastuseks sellele pööratakse tihtipeale interneti poole, kust üritatakse saada võimalikult palju lisateadmisi võimalikult lihtsalt ja lühikese ajaga. Paraku on internet informatsioonist üle kuhjatud, mis teeb kindla asja leidmise raskeks. Lisaks sellele on raske leida ühtset tervikliku rakendust arusaadava struktuuriga ning tihtipeale peab otsima juppide kaupa ning nendest panema kokku ühtse pildi.

Käesoleva lõputöö eesmärk on luua veebipõhine massikursuse rakendus, mis hoiaks endas kursuseid igast võimalikust valdkonnast. Lahendusest saavad abi nii õpilased, tudengid kui ka need, kes soovivad enda teadmisi täiendada ja olla tööturul konkurentidest ühe sammu võrra ees. Teiselt poolt on rakendus kasulik ka inimestele, kes soovivad oma teadmisi ülejäänud maailmaga jagada aidates teisi ning samuti saada kogemust ja lisateadmisi õpetamise läbi.

Veebirakendus on loodud kasutades REST (*Representational state transfer*) API (*Application Programming Interface*) arhitektuuri. Ülesehitatud on see kasutades klient-server mudelit, mis viitab sellele, et rakendusel on eraldiseisvad kasutajaliides ja serveripoolne liides, mis omavahel suhtlevad ning edastavad üksteisele andmeid. Graafiline liides on loodud võimalikult lihtsalt ning kasutajale arusaadavalt, et ka vanema generatsiooni inimesed, kellel üldjuhul arvutiteadmised pole nii tasemel, suudaksid lehel võimalikult kergelt navigeerida. Serveri pool aga hoiab endas kogu rakenduse funktsionaalsust ning suhtlust andmebaasiga.

Käesoleva lõputöö raames uuritakse ka põhjalikumalt problemaatikat, miks on sarnaseid rakendusi tänapäeva ühiskonnas vaja. Samuti tuuakse välja ka alternatiivid, mis tänapäevaseks on loodud ning analüüsitakse nende tööpõhimõtet, tugevusi ja nõrkusi ning

erinevust antud projektist. Lisaks on toodud välja ka rakenduse võimalikud lisarakendamise võimalused pikemas perspektiivis.

Teine peatükk keskendub peamiselt vahendite valikule. Tänapäeva rakenduse loomine koosneb üldjuhul mitmest etappist ning pea igaühe jaoks on olemas eraldi programm või rakendus, mis suudab selle võimalikult lihtsalt ja efektiivselt ellu viia. Peatükis räägitakse iga selle etappi valitud vahendist, analüüsitakse alternatiive ning põhjendatakse detailsemalt valiku tegemist ning kriteeriume, mille alusel oli valik tehtud.

Viimaks jääb programmi enda arendamine ning selle detailne kirjeldus. Põhjendatakse igat arendamise taset, mida tehti ning miks nii tehti. Jooniste näitel tuuakse välja ka näiteid rakendusest ning koodist, mis selle võimalikuks tegi. Lõpuks on kirjeldatud ka lõpptulemust, mis näitab loodud veebirakenduse funktsionaalsust ning võimalusi, mis seal on võimalik teha.

1 Problemaatika

Tänapäeva ühiskonnas on konkurents tööturul muutunud palju tihedamaks. Peamisi põhjusi on see, et nooremad generatsioonid võtavad haridust palju tõsisemalt ning uuringud on näidanud, et 2016. aasta andmetel omab ligi 40% Ameerika Ühendriikide töötavast Y-põlvkonnast (25-29 aastased) bakalaureuse või veel kõrgemat hariduskraadi. Antud arv on kasvanud viimase 50 aastaga umbes 24%, mis näitab, et vaatamata sellele, et haridus on muutunud kallimaks viimaste aastate jooksul on see noortele tänapäeval palju tähtsam kui ka kergemini kättesaadav arvestades tehnoloogia ja ühiskonna arengut. [1]

Olles üldiselt igati positiivne nähtus, tekitab see omakorda tööturul palju tihedama konkurentsi. Ettevõtted ei võta enam arvesse üksnes haridustaset, vaid uurivad ka individuaalide lisateadmisi ning kogemusi, mis annaksid parema ülevaate kandideeritava oskustest. [2]

Tudengid peavad seetõttu tegema paraja koguse lisatööd kooli õppeainete kõrvalt, et olla konkurentidest alati üks samm eespool. Olgu selleks näiteks projekt, mis näitab nende teadmisi ning oskusi antud valdkonnas või erialal, või lisakeele õppimine, mida tihti tahetakse või otsitakse ettevõtete poolt. Üldjuhul tehakse seda kõike kas raamatute abil või abiõpetaja teenuseid kasutades, mis tänapäeval paraku ei ole kõige efektiivsem lahendus.

Raamatutega on alati risk valida vale raamat, mis ei pruugi olla piisavalt efektiivne ning ei anna vajaliku infot võimalikult kergelt, kiirelt ja arusaadavalt edasi. Samuti ka nende kogus teeb valiku palju raskemaks ning osad tehnoloogia või isegi keelte raamatud võivad olla ka aegunud, arvestades kui kiire areng tänapäeva ühiskonnas on ja kui kiirelt muutused toimuvad.

Abiõppejõu puhul teeb problemaatiliseks asukoht ja keelebarjäär. Olles näiteks Eestis, kus rahvaarv vaevu ületab 1,2 miljoni elaniku, on võõrkeele õpetajat raskem leida, kui kuskil internatsionaalsemas riigis ja suurema rahvaarvuga nagu näiteks Inglismaa või

Holland. Ka viimase aasta sündmused on näidanud, kui oluline on võimalus hoida ühendust abiõpetajaga ka elektroonilisel teel.

Piisava vedamise ja tahtejõu korral on võimalik ka neid meetodeid kasutades omandada lisateadmisi, kuid praktika on näidanud, et tänapäevaks on loodud palju efektiivsemaid võimalusi saada kogemust ning lisateadmisi. Üheks selliseks oleks antud lõputöö raames loodud veebipõhiste kursuste veebirakendus, mis enamik välja toodud puuduseid likvideeriks.

1.1 Kursuste veebirakendus

Käesoleva lõputöö raames loodud projekt on REST API arhitektuuril loodud veebipõhine massikursuse veebirakendus. Rakendus on mõeldud lahenduseks eelnevalt mainitud probleemidele. Igal kasutajal oleks vaba ligipääs veebilahendusele, kust nad saaksid valida endale sobivama e-kursuse endale sobival teemal. Samuti oleks võimalik kursuseid filtreerida kategooriate järgi, mis aitaks kitsendada valikut teatud valdkondadele. Täpsemate kursuste leidmiseks oleks kasutajal sisseehitatud ka dünaamiline otsingusüsteem, mis leiaks kursuseid nende nimes sisaldavate tähtede järgi. Lõpuks oleks võimalik kitsendatud valikut ka sorteerida kas hinnangu, hinna või pikkuse järgi.

Nagu sai mainitud, oleks tegemist REST arhitektuuri kasutatava rakendusega. See tähendab, et veebilahendusel on eraldi seisvad kasutajaliides ja serveripoolne liides. Suhtlus toimub kasutaja poolt saadetud päringute abil. Server saab kasutajalt päringu ning vastavalt sellele teeb vajalikud toimingud andmebaasis ning tagastab JSON (*Javascript Object Notation*) objekti. Kasutajaliides aga võtab vastu tagastatud JSON kujul objekti ning kuvab selle kasutajale läbi veebilehe ekraanile arusaadaval ja ilusamal kujul. [3]

Rakendust saaksid kasutada kahte tüüpi kasutajad. Üks pool, kes saaks kasu oleksid tudengid või õpilased, kes vajavad, kas abimaterjali seoses kooliainetega või -teadmistega või lihtsalt otsivad lisateadmisi, kas mingis kindlas töövaldkonnas, keelekursuseid või ka näiteks mingi hobikursuseid. Viimased kolm oleksid kasulikud kindlasti ka tavalistele inimestele, kellel kool on läbi ning töökoht olemas, kuid soovivad õppida midagi juurde, et käia maailmaarenguga kaasas ning olla tööturul konkurentidest alati ühe sammu võrra ees. Teine pool hõlmaks endas aga nii-öelda õpetajaid, inimesi, kes soovivad teisi aidata

mingi teema või uue oskuse õppimises. Nemad vastutaksid enda kursuste lisamise eest antud veebilehele.

Eesmärgiks oleks luua rakendus, mis eemaldaks kõik või vähemalt enamus olemasolevate e-õppe lahenduste vigu. Projektil oleks olemas süsteem, oleks kerge kasutada ning kõigile kättesaadav. Informatsioon oleks võimalikult ajaga kaasas käiv ning veebilehe ja kursuste struktuur arusaadav.

1.2 Olemasolevad rakendused

Lisaks raamatutele ja personaalsetele abiõpetajatele on olemas ka kolmas abikäe pakkuja, milleks on internet. Selle puuduseks on selle suurim eelis - informatsiooni üleküllus. Tihtipeale võib internetist leidmine võtta kauem aega, olenevalt kui spetsiifiline otsitav teema on. Tagajärg võib olla see, et mitmest väiksest osakesest tuleb panna kokku üks suur tervik, mis järjekordselt võtab aega ning seetõttu ei ole just kõige efektiivsem lahendus.

Tänapäevaks on loodud mitmesuguseid alternatiive lõputöö käigus valmivale rakendusele, millel kõigil on nii omad plussid, mida käesolev rakenduse arendusel oli üritatud jäljendada. Paraku lisaks rakenduse tugevale küljele leidub ka negatiivseid külgi, mida antud lahendus on üritanud mitte korrata ja luua kasutajatele paremad võimalused.

UDEMY on tänaseks päevaks muutunud üheks suurimaks lisakursuste pakkujaks ning tõenäoliselt oleks ka tööturul suurimaks konkurendiks. Udemy sisaldab endas napilt üle 150 tuhande kursuse, mis on loodud ligi 70 000 õpetaja poolt 65-s erikeeles. Veebilehel leiab erinevaid kategooriad, mille järgi saab igat kursust valida ja filtreerida enda meelpära järgi. [4] Miinuseks on aga välja toodud seda, et Udemy annab väheste kursustega välja ka sertifikaadi, mis omakorda ei ole nii oluline kui haridustase, kuid näitab, et potentsiaalne kandidaat on panustanud lisatöösse ning omab mingis teatud valdkonnas lisateadmisi ja oskusi, mis võivad tulla kasuks. Ka kursuste kontroll ei ole just kõige parem, võttes arvesse kui suur hulk erinevaid kursuseid seal leidub. Osad lisatunnid võivad olla puudulikud, sisaldada vale infot või olla koguni teemast mööda. Olenevalt sellest, mis eesmärgiga klient veebilehte külastab võib Udemy struktuur ja ülesehitus olla nii puudus kui ka eelis. Puudusena võib see tulla nende jaoks, kelle eesmärk on leida võimalikult kiirelt võimalikult täpselt tehtud kursuse, sest nagu eelnevalt

sai mainitud, sisaldab Udemy tuhandeid kursuseid erivaldkonnas ning ühe kindla ja sobiliku leidmine võib olla keeruline. Eeliseks on see aga neile, kes otsivadki mitmeid üksteisest erinevaid kursuseid. [5]

COURSERA oma eesmärgi ja põhimõte poolest küll erineb, omades õpetajate rollis peamiselt ülikoole ja organisatsioone, aga sellegi poolest on see üks suuremaid antud teenuse pakkujaid üle terve maailma. Coursera kursuseid kasutab või on kasutanud ligi 77 miljonit inimest ning on partnerluses 200 tippülikooli ja ettevõttega. Erinevalt eelmainitud Udemyst, pakuvad Coursera lisakursused ka sertifikaate, mis on tööturul paljude organisatsioonide poolt kõrgelt hinnatud ning on seetõttu Udemys üks samm ees. [6] Miinuseks tuleb see rohkem, aga inimestele, kellel paraku pole võimalik enda teadmisi teistega jagada ning olemasolevatest kursustest paremaid üles laadida, sest seda teevad organisatsioonid ja ülikoolid. Sellest tingituna tekib ka asjaolu, et osade kursuste informatsioon on aga ajast maas ning vajab uuendamist, eriti tehnoloogia valdkondades, kus areng liigub kiirelt edasi ning ei haridusasutused ega ka ettevõtted ei jõua sellele järgi, sest on tihtipeale harjunud juba mingite kindlate programmide ja lahendustega, mida peamiselt ka kasutatakse. [7]

YOUTUBE on maailma suurim video platvorm, kus inimesed postitavad igapäevaselt enda videoid täiesti erinevatel temaatikatel. Siin keskkonnas leidub ka inimesi, kes tahavad enda teadmisi jagada teistega ning postitada vastavaid abistavaid videokursuseid. Youtube suurimaid eeliseid teiste alternatiivide ees on, et see on maailma teine kõige enim kasutatavamaid veebilehti maailmas ehk tõenäosus, et õpetus jääb kellelgi silma on mitu korda suurem, kui teiste alternatiivsete lahenduste puhul. [8] Samuti ei tohi unustada, et videote vaatamine on selles keskkonnas tasuta ehk õpilased ei pea mitte midagi maksma ega ka tegema mingit kasutajat selleks, et antud rakenduse kõiki võimalusi ära kasutada. Puuduseks aga teiste alternatiivide ees on muidugi juba Udemy puhul mainitud sertifikaadi saamine, mis tähendab, et kõik videokursused on mõeldud täielikult inimese enda arengu jaoks. Asjaolu aga, mistõttu antud lahendus jääb samuti enamustele lisakursuseid pakkuvatele rakendusetele alla on struktuuri puudumine. Erinevalt eelmainitud lahendustest, puudub Youtube-il nii-öelda faktikontroll, mis kontrolliks videokursuse vastavuse teemale, mille jaoks see on loodud. Puudub jätk ning põhjalik üldisema temaatika selgitus, mis aitaks valitud ala paremini mõista või seostada. Ka omandatud teadmisi on tihtipeale raske kontrollida, sest üldjuhul puudub korralik

tagasiside võimalus. Kommentaarid, eriti videotel, kus on palju vaatamisi, jäävad tihtipeale vastamata või üldsegi vaatamata. [9]

INTERNETIS leidub miljoneid erinevaid artikleid, lehti, veebirakendusi ükskõik mille ja mis temaatika kohta. Antud hetkel loetakse alternatiivina kõiki väiksemaid veebilehti, mis on loodud mingi kindla teema õppimiseks või lahendamiseks. Samuti on need lehed tihtipeale mõeldud vaid ühele valdkonnale. Näiteks Codeacademy, mille eesmärk on õpetada inimesi programmeerima ning viia kurssi IT-maailmaga viies läbi erinevaid lühikursuseid ja lahendades ülesandeid. [10] Ka erinevad veebilehed, mis räägivad muusikateooriast või näiteks klaverile omaseid teemasid. Sarnaselt Youtube lahendusele, puudub ka antud olukorras üldjuhul igasugune struktuur, sest terviku peab tihtipeale panema kokku mitmes väiksemast osakesest kokku. Samuti enamus lehti on üldsõnalised ning annavad ülevaadet vaid baasteadmistele. Raskemate ja spetsiifilisemate olukordade uurimine on raskem, sest nõuab kindlamat arusaama, mida kasutaja otsima peab, kuid struktuuri puudumise tõttu puudub üldjuhul ka vastavad teadmised, mis järgmine etapp olema peaks.

MENTORNAUT on tõenäoliselt üks ainukesi veebilahendusi Eesti maastikul, mis oleks lähedal käesoleva lõputöö projektile. Tegu on startup ettevõttega, mis pakub kooliõpilastele lisatunde ja kursuseid, mis aitavad kooliainetes ja valmistumisel eksamiteks. [11] Mentornauti erinevus seisneb selles, et kursused ja tunnid on mõeldud vaid kindlale inimeste grupile ning õpetajateks on üldjuhul tudengid või päris õpetajad. Antud projekti eesmärgiks on aga võimaldada kõigil proovida käsi õpetamises ning jagada enda teadmisi ja suunata ka valdkond rohkem hobide poole ja grupirühm tudengite või vabatahtlike poole. Mentornaut oleks Eesti maastikul suurimaid alternatiive, eriti just õpilaste seas, kuid mis puudutab vanemaid generatsioone, siis antud hetkel lähevad antud projekti ja startupi eesmärgid lahku.

1.3 Võimalikud lisarakendamised

Mitmeid alternatiive uurides, tuli ilmsile ka üks oluline rakendusala, mida väga vähesed teised lõputöö projekti sarnased lahendused kasutavad, kuid mis oleks silmapaistev eelis teiste ees. Selleks on koostöö ülikooli ja erinevate organisatsioonidega, kes pakuvad enda töötajatele õppimisvõimalusi. Üheks selliseks on näiteks Coursera. [6]

Enamus ettevõtetel on tänaseks olemas õppeplatvorm, kus nad viivad oma töötajatele läbi erinevaid lisakursuseid, millest osa on kohustuslikud kõigi jaoks ning teine hulk aga vabatahtlik ning enesearendamiseks mõeldud. Olles uurinud osasid õppeplatvorme organisatsioonides, on sealsed lahendused tihtipeale üpris vanamoodsad ning vigaste ning eksitavate kasutajaliidestega. Samuti ka veebilehe struktuur ei ole kõige arusaadavam ning kasutamine on raskem kui tavalistel sarnastel rakendustel. Antud lõputöö käigus välja arendatud lahendus parandaks enamik neid probleeme ära ning pikemas perspektiivis oleks seda võimalik ühendada ning kui vajalik siis teha ka ümber vastavalt organisatsiooni nõuetele, mis peamiselt puudutaks veebilehe kliendipoolset osa.

Integreerimine oleks tulevikus võimalik ka koolisüsteemidega, mis aitaks välja mitmeid tudengeid ja õpilasi, kellel on igasuguseid raskusi mingite ainete või kindlate teemadega. On juhtunud kui õppejõud ei paku tudengitele konsultatsioone või nendest ei ole osadele ülikooli õpilastele piisav ning pööratakse oma kursusekaaslaste poole. Üldjuhul jäävad sellised koosõppimised väiksele ringile ning osad vähemaktiivsemad inimesed jäävad sellest ilma ning ei saa teatud teemasid selgeks. Juhul kui oleks olemas portaal kuhu taibukamad tudengid saaksid mingil teatul teemal panna üles enda tehtud abikursuseid, saaksid mitte ainult kõik abivajavad rühmakaaslased kasu, vaid ka ülejäänud kursusekaaslased erinevalt erialalt. Samuti tuleks see kasuks ka järgmisetele põlvkondadele niikaua kuni antud aine või teema veel aktuaalne on.

1.4 Järeldused analüüsist

Maailmas on olemas mitmeid alternatiivseid lahendusi, mis pakkuvad lisakursuste võimalusi, mis täiendavad teadmisi ning annavad ka kogemust. Enamustel neist puudub struktuur ning on mõeldud vaid üksikute küsimuste vastamiseks või väikeste probleemide lahendamiseks. Samuti enamik lehti on mõeldud vaid kindlale valdkonnale/erialale, eelkõige tehnoloogiale. Puudub ühtne platvorm näiteks hobidele ja humanitaaraladele. Näiteks platvorm, kus oleks peamiselt võimalik õppida erinevaid muusikainstrumente, joonistamistehnikaid, fotograafiat ja ka keeli. Olulisim peaks selle kõige juures olema struktuur, abitunnid ei tohiks sisaldada tühja ja aegunud infot, samuti selgitused peaksid olema piisavalt täpsed ja arusadavad auditooriumile. Lahenduses ei tohiks olla ühe teema kohta kümneid või isegi sadu erinevaid samasuguseid kursuseid, mis kokkuvõttes annavad ühte ja sedasama edasi.

Alternatiivide juures tuleks pöörata tähelepanu ka sellele, et enamus on rahvusvahelised veebirakendused, kus peamiseks õppekeeleks on inglise või mõni muu laialdasemalt levinud keel. Küll aga Eesti ühiskonnas sellised rakendused, kus ka eesti lapsed, õpilased, tudengid saaksid õppida endale südamelähedast ainet emakeeles, on puudulikud või on neid väga vähe ja üldjuhul halva ülesehitusega ning läbimõtlemata.

Käesoleva lõputöö käigus valmis rakendus elimineeriks kindlasti enamus alternatiivide juures välja toodud vigu kui ka oleks kasutajatele kerge ja arusaadava ülesehituse ja välimusega. Korraliku filtreerimise ja sorteerimise meetoditega, mis aitab leida otsitava kursuse kiiremini, kui näiteks läbi interneti või Youtube. Pikemas perspektiivis oleks kindlasti vajalik tugevat postitatud kursuste kontrolli, mis hoiab ebavajaliku ja tühja informatsiooni postitamisest.

2 Vahendite valik

Antud lõputöö praktiliseks väljundiks oli luua hajus veebirakendus. Lahenduses peab olema võimalik isiklike kursuste lisamine, meeldivama valimine ning tagasiside jätmine. Rakendus pidi olema ülesehitatud REST API arhitektuuril kasutades klient-server-mudelit. Omakorda tähendab see seda, et veebilehel on olemas mootor, mida üldjuhul nimetatakse serveripoolseks liideseks ehk *back-end*-iks, kus toimub kogu suhtlus andmebaasiga ning kogu funktsionaalsus on ehitatud seal. Klient aga näeb ja pääseb ligi vaid kliendipoolsele ehk graafilisele liidesele, mida inglise keeles kutsutakse ka *front-end*-iks. Seal on peamiselt tegeletud välimuse ning kasutatavusega.

Tänapäevaks on välja arendatud hulgaliselt programme, programmeerimiskeeli, andmebaasi servereid ja kõike muud, mida antud projekti valmistamiseks vaja läheks. Valiku tegemisel pöörati peamiselt tähelepanu sellele kui laialdaselt kasutatav on antud programm mujal maailmas, kui paljud rakendused on selle peal ehitatud ning mitmed suurettevõtted seda lahendust kasutavad. Arvesse oli võetud ka mugavus ja lihtsus, mis sellega kaasas käis ehk palju ja kui kergelt leiab informatsiooni antud programmi kohta näiteks internetis. Samuti peeti silmas ka seda, et vahendid ei oleks liiga kauged ülikooli jooksul õpitust ning omandamisele läheks minimaalselt aega kui üldse.

2.1 Projektijuhtimise tarkvara

Tänapäeva ühiskonnas saab iga projekt alguse sellest, et selgitatakse välja eesmärgid, mida antud lahendus tegema peab. Luuakse plaan, kus pannakse paika ka kindlad tähtajad ja ülesanded, mille ümber hakkab käima ka kogu projekti arendus. Tehakse seda kõike eeposte ehk *epic*-ute ja probleemide ehk kasutajaloode näol.

Kui varem olid eelnevad punktid pandud kirja kas käsitsi paberile või oli spetsiaalselt loodud exceli tabelid, siis aja pikku on selle jaoks loodud mitmeid tarkvaralisi lahendusi, mis teevad kogu projektijuhtimise ning töövoo monitoorimise palju mugavamaks ning efektiivsemaks. Üheks suurimaks antud valdkonna rakenduseks peetakse Jira tarkvara, mida oli kasutusele võetud ka käesoleva töö projekti loomisel.

Antud valik oli tehtud juba eelnevalt projekti alustamisele seetõttu, et veebirakenduse arendamine oli alustatud Kuehne+Nagel-i ettevõttepraktika raames ning mainitud organisatsiooni esmane projektijuhtimise tarkvara on Jira. Kasutatud lahenduse eelisteks mängib asjaolu, et see on üks enimkasutatavaid sellelaadseid rakendusi maailmas, üle 65 000 tiimi/firma on võtnud selle endale kasutusele, mis tõstab ka tööturul selles oskustega töötaja teistest kõrgemale kohale. [12] Vaatamata selle populaarsusele ei ole Jira algajate sõbralik ning omab keerulist ja kohati segadusse ajavat kasutajaliidest, mis on ühtlasi ka vananenud. See on mõeldud peamiselt suurtele organisatsioonidele, kus on palju erinevaid keerukaid süsteeme. Mis aga puudutab iseseisvaid väiksed projekte, siis on loodud mitmeid alternatiive, kus nii graafiline liides kui ka funktsionaalsus on kerge ning arusaadav ja sobib imeliselt ka algajatele. [13] Kaks enimkasutatavat tehnoloogiat on ClickUp ja Trello. Trello on loomuselt väga sarnane Jirale, kuid palju efektiivsem ning arusaadavama disainiga. Hetkel on see ka üks enimakasutatavaid Jira alternatiive ning on pakkunud viimaste aastate jooksul kõva konkurentsi. [13] [14]

Teine eelnevalt mainitud alternatiiv on ClickUp, mille näol on tegemis hetkel üheks parimaks Jira alternatiiviks peetava rakendusega. Tegu on samuti vägagi lihtsalt ülesehitatud kasutajaliidesega, mis on nii algajasõbralik kui ka professionaalne, pakub seejuures ka hulk lisavõimalusi. ClickUpi on võimalik kasutada ka mitte ainult kui projektijuhtimis rakendusena, vaid ka kui tavapäeva planeerijana. Sarnaselt Jirale on ka ClickUpil kolmanda osapoolte rakendustega integreerimine. Enimkasutusel olevatest sellistest rakendustest on GitHub ja Slack, mis toob ClickUpi kõikidest teistest Jira alternatiividest oluliselt välja, kaasaarvatud ka Jirast endast. [15]

Jira teadmine ja selle kasutamise oskus on tänapäeval veel oluline ning kindlasti tuleb tööturul kasuks, sest on siiani väga paljudes firmades peamiseks projektijuhtimise rakenduseks. Küll aga väikeste iseseisvate projektide puhul soovitatakse kasutada ja õppida vähemkasutusel olevaid kergemaid alternatiive nagu ClickUp või Trello, mis on kaasaegsemad ja loomu pooles arusaadavama struktuuri ja ülesehitusega.

2.2 Back-end ehk serveripoolne liides

Back-end-i ehk serveripoolset liidest peetakse iga rakenduse mootoriks. Selle sisu on veebirakenduse kliendile üldjuhul peidetud ning kogu funktsionaalses ning toimingud toimuvad seal. Paralleeli võib tuua autoga, kus front-endiks on nähtaval olevad esemed

nagu näiteks pedaalid, rool ja käigukast, kuid kogu nende funktsionaalsus ning tööpõhimõte on nii juhi kui ka kaassõitjate poolt peidetud üldjuhul kapoti all, kus asub nö. auto *back-end*. [16]

Serveripoolses liideses tehakse peamiselt tööd andmebaasi ja serveriga ning kogu mootor on ehitatud mingisuguse programmeerimisekeelega kasutades mingit kindlat raamistikku. Eeltoodud komponentide valik on tänapäeval tehtud väga suureks, mis ühelt poolt teeb valiku kergeks, sest päris tõenäoliselt leiab sealt midagi sobilikku. Teiselt poolt on aga valiku tegemine raske, sest pea võimatu on võtta välja üks lahendus ning öelda, et see on kõige parem. Seetõttu oli otsustatud valida ülikooli vältel õpitud rakenduste seast.

2.2.1 Programmeerimiskeel ja raamistik

2020 aasta andmetel on maailmas loodud üle 700 erineva programmeerimiskeele. Isegi kui eemaldada sellest hulgast kõik vananenud keeled, nagu näiteks Lisp ja igasugused esoteerilised ja harvakasutatavad keeled ning jätta alles vaid enimpopulaarsemad ja maailmas kasutusel olevad, jääks ikkagi valikusse ligikaudu viiskümmend erinevat programmeerimiskeelt. [17]

Nagu juba sai eelnevalt mainitud, nendest viiekümnest populaarseimast kitsenes antud tööprojekti arendamisel valik vaid ülikoolis õpitud keelte peale. Nende vahel valides keskenduti peamiselt sellistele kriteeriumitele nagu:

- Kui aktuaalne üks või teine keel on ehk kui populaarne on keel tööturul?
- Kui suur kogemus väljatoodud programmeerimiskeelega on ehk kui palju peaks juurde õppima?
- Kui kerge ja ühtlasi ka sobilik antud keel selle veebirakenduse arendamiseks on, kas selle jaoks on loodud piisavalt abivahendeid ja mooduleid?

Ülaltoodud küsimusi kasutades oli loodud tabel iga õpitud programmeerimiskeele analüüsimiseks. Igat veergu hinnati kümnepallisüsteemis, kus null sümboliseeris väga positiivset ehk näiteks esimese küsimuse näitel väga aktuaalne ja kümme väga negatiivne ehk sama näite puhul, üldse mitte aktuaalne.

Tabel 1. Programmeerimiskeelte võrdlustabel.

	PHP	Java	C#	Python	Javascript
Aktuaalsus	7	0	3	4	1
Kogemus	5	2	1	6	3
Lihtsus	4	1	2	3	3
Kokku	16	3	6	13	7

Aktuaalsust hinnates oli võetud hetkel maailmas enimkasutatavate keelte edetabeleid, kui ka Eesti tööturgu. Väljatoodud keeltest tuli kindlalt võitjaks Java. Põhjuseks, et see on üpris universaalne ning võimaldab teha mitmeid erinevaid lahendusi. [18] Vaatamata sellele, et Python ja Javascript antud hetkel pole maailmas nii üldtunnustatud ja kasutatavad keeled, olid nad mitmes edetabelis toodud välja kui kiirelt arenevad ning tulevikukeeled, millel on potentsiaali kuulsust kasvatada. [19] Ka see oli keelte puhul arvesse võetud, kuid sellele vaatamata jäi Java siiski antud kategoorias võitjaks. PHP, mida küll surnud keeleks veel ei peeta, hakkab sellegi poolest oma populaarsust kaotama ning jääb alla enamustele tänapäeval kasutuses olevatele uuematele keeltele. [20]

Lihtsuse puhul tuli samuti võitjaks Java, peamiselt tänu sellega kaasaskäivale Spring raamistikule, mis muudab veebiarenduse mugavamaks ja kergemaks ning koodi ja selle struktuuri arusaadavamaks. Lisaks sellele ka informatsiooni ja dokumentatsiooni leidmine internetis on Java kohta väga kerge. Javale on välja arendatud ka erinevaid lisamooduleid, mis muudavad arendatava rakenduse turvalisemaks, stabiilsemaks ning selle programmeerimise mugavamaks ning lihtsamaks.

Lisades eeltoodud kahele näitajale viimase hinde, milleks oli käesoleva rakenduse arendaja isiklik kogemus antud programmeerimiskeeltega, tuli kokkuvõttes valikuks Java.

Javal on ka teisi funktsionaalsuse eeliseid teiste keelte ees. Üheks selliseks on näiteks keele universaalsus. Java rakendus on platvormist sõltumatu. See tähendab seda, et

rakendust võib ilma mingisuguse raskuseta viia üle ühest süsteemist teise. Samuti on ka süsteemisisesed uuendused ei mõjuta Java rakenduse toimimist. [21]

2.2.2 Andmebaas

Serveripoolses liideses toimub suhtlus andmebaasiga, mida peetakse kogu rakenduse ajuks. *Back-end* peab reageerima igale kliendi poolt tehtud päringule ning vastavalt sellele edastama päringu andmebaasile ning tagastama vastava informatsiooni. [16] Seetõttu õige andmebaasi valik on samuti väga oluline iga veebirakenduse arendamisel.

Valiku tegemisel ei võetud arvesse andmebaase, mis on mõeldud suurtele süsteemidele ning lisaks kõigele on ka tasulised. Üheks selliseks oli näiteks Oracle andmebaasi serverid. Samuti ei arvestatud ka SQLite andmebaasi, mis on oma funktsionaalsuse ja mahu poole pealt limiteeritud. Lähtuti peamiselt olemasolevast kogemusest õpitud andmebaasidega.

Lisavalik oli tehtud ka arenduse ajaks, kus oli vaja kiiret ja kergekaalulist andmebaasi, mis võimaldas teha pea kõike, mida ka üldkasutatavad serverid võimaldavad. Selleks osutus H2 andmebaas. Mainitud andmebaas on ehitatud Javas ehk seda on väga kergesti võimalik integreerida ükskõik millisesse Javaga ülesehitatud rakendusse. Oma kiiruse, lihtsuse ja mugavuse poolest on antud andmebaas eriti hea just arendamise protsessis, kui tegu on väikeste vähemkeeruliste süsteemidega. Pikemaks perspektiiviks antud andmebaasi ei soovitata, sest üldjuhul kustutatakse andmed iga rakenduse käivitamise tagant. [22]

Teise andmebaasi mootorina oli valitud Microsoft SQL server ning peamiseks põhjuseks jäi kogemus. Tegu on samuti kiire ja funktsionaalse andmebaasiga. Väikeste süsteemide puhul saab andmebaasi kasutada tasuta, mis paraku ei käi suuremate ja keerukamate süsteemide kohta. [23]

2.3 Front-end ehk graafiline liides

Graafiline liides ehk kasutajaliides on veebirakenduse välimus, mida klient ka ise näeb. Seda võib selgitada kasutades juba eelnevalt mainitud näidet autoga. Auto niinimetatud *back-end* oli meil välja selgitatud, see oli mootor ja kõik muu, mida juht ei näe. *Fron-*

end-ks võib aga lugeda kõike, mida juht näeb ning saab kasutada, näiteks pedaalid, käigukast ja rool.

Kasutajaliidese arendus hõlmab endas rakenduse välimise struktuuri arendamist kasutades HTML-i (*HyperText Markup Language*), CSS-i (*Cascading Style Sheets*) ja Javascripti, mis lisab lehele dünaamilisust ja funktsionaalsust. Kasutajal on võimalik juhtida front-endi, mis omakorda suhtleb serveriga saates kasutaja poolt tehtud päringuid. Server omakorda edastab päringud andmebaasi, teeb vastavad toimingud ära ning saadab tulemuse tagasi graafilisele liidesele, mis kuvab omakorda kasutajale. [24]

Nagu eelnevalt sai välja toodud, käib front-end arendus üldjuhul Javascriptiga. Küll aga on tänapäevaks loodud hulk raamistike nii tavalise kui ka tüüpidega Javascripti ehk Typescriptile.

2.3.1 Typescript

Enda loomu poolest on Typescript tavaline Javascript, see algab sellega ja lõppeb. See tähendab seda, et iga Typescripti rakenduse jooksutamisel muutub masinkood automaatselt Javascriptiks tagasi ehk Typescript kui selline eksisteerib vaid arenduse käigus. [25]

Typescripti üks peamisi erinevusi on see, et kõik on tüübitud, samamoodi nagu seda on ka Javas, C# ja peaaegu igas teises C-tüüpi programmeerimiskeelel. Eeliseks tuleb see, et vigade tegemise tõenäosus on selle võrra väiksem, sest igal pool seisab kontroll ning on teada, mis tüübiga tegu on. Typescript omab endas kõike, mis on Javascript ja isegi enam väheste lisade näol. Programmeerija jaoks aga kõige olulisem erinevus seisneb koodide kompilieerimisel. Javascripti puhul tuleb viga välja alles siis, kui rakendus on jooksutatud ning viga tekkimisel peab arendaja üldjuhul seda ise taga ajama. Typescripti puhul toimub vigade otsimine kompileerimise ajal, mis kiirendab vea leidmise protsessi isegi vaatamata sellele, et Typescript kompilaatorisse on sisseehitatud veaotsimis mehhanism, mis näitab ise täpselt ära, kus ja mis real viga tekkis. [26]

Tuginedes väljatoodud materjalile, oli Typescript valitud ka käesoleva lõputöö projekti graafilise liidese keeleks. Vaadates ka maailma edetabeleid on näha, et Typescripti populaarsus on iga aastaga kasvamas, mis toob esile ka selle aktuaalsuse tööturul.

Lisapunktidena võib lugeda ka antud veebirakenduse arendaja kogemuse ning keele turvalisuse ja vigade tekkimise väikese tõenäoususe.

2.3.2 Raamistik

Kasutajaliidese programmeerimisel mängib olulist rolli Javascripti raamistik, milles antud rakenduse koodi kirjutatakse. Tänu neile on enamus tänapäeva rakendusi meeldiva välimuse ja tunduvad kasutajale loomult lihtsalt, vaatamata sellele, et tagaplaanil toimub ohtralt keerukaid funktsioone. [27]

Raamistiku valimiseks oli kasutusele võetud Tabel, kus analüüsiti kolme suurimat Javascripti raamistiku, mida hetkel kasutatakse: React.js, Angular.js ja Vue.js. Analüüsimiseks oli püstitatud kriteeriumid, mis aitasid rakendusi omavahel paremini võrrelda ning valiku langetada.

- Kogemus valitud raamistikuga.
- Populaarsus tööturul.
- Raamistiku sobivus antud projekti lahendamiseks.
- Dokumentatsiooni ja informatsiooni leidmise võimalused internetis.

Sarnaselt programmeerimiskeele valikule, toimis ka siin reegel, mida väiksem number, seda sobilikum on raamistik käesoleva rakenduse ehitamiseks.

Tabel 2. Javascript raamistike võrdlemise tabel.

	React.js	Angular.js	Vue.js
Populaarsus	0	0	1
Sobivus	1	4	2
Kogukond ja dokumentatsioon	2	1	5
Arendaja kogemus	1	6	4
Kokku	4	11	12

Populaarsuse juures said kõik kolm pea maksimaalse punkti, sest on maailma edetabelis juhtivatel positsioonidel. Teiste eeliseks Vue ees oli see, et Vue ei ole nii laialdaselt kasutusel ei Ameerika Ühendriikides ega ka Eesti pinnal.

Projektile kõige sobivamaks ostus React.js. React-il on võimalik taaskasutada komponente, mis vähendab koodi ja failide tegemist süsteemis, ühtlasi vähendades ka rakenduse keerukust. Lisaks sellele on nii React kui ka Vue mõeldud nii suurtele korporatsioonide süsteemide jaoks kui ka väikestele individuaalprojektidele, mis antud hetkel olukorras ka nende kasuks. Angular seevastu omab tohutult lisasid, mis teevad selle perfektseks valikuks suurte ja keerukate rakenduste puhul, kuid mittevajalikuks selliste väiksemate rakenduste puhul, sest suurem osa selle funktsioonalsusest ja pakutavatest võimalustest jääb realiseerimata. [27]

Kogukond ja dokumentatsiooni puhul oli arvestatud mõlemat korraga. Näiteks nii Angularil kui ka Reactil on olemas põhjalikud dokumentatsioonid kas raamistike arendajatelt või kolmandatelt osapooltelt. Lisaks ka kogukond on märkimisväärselt suur, mis aitab informatsiooni ja abi kergemini ja kiiremini leida. Vue puhul aga, vaatamata sellele, et arendaja poolt on loodud üpriski silmapaistev ja põhjalik dokumentatsioon, puudub sellel nii suur kasutus kui ülejäänud kahel raamistikul, mis omakorda näitab, et üldsus puudub ning abi internetis on leida kordades raskem. [27]

Liites eelnevatele tulemustele juurde ka varasema kogemuse, mis kaasab endas ka raamistiku lihtsuse [27], osutus valikuks React.js raamistik. React on lihtne raamistik, mis sisaldab endas kõike vajaliku graafilise liidese arendamiseks. Komponendi on taaskasutavad ning kood ei ole ülesehitatud klassidel, vaid niinimetatud React *hook*-idel, mis hõlbustab nii kirjutamise kui ka õppimise ning muudab protsessi palju meeldivamaks. [27]

3 Rakenduse loomine

Peatüki esimeses pooles toimub rakenduse teoreetiline ettevalmistus. Luuakse vastavad kasutajalood ja *epic*-ud antud rakendusele Jira keskkonnas ning pannakse paika tähtajad, raskusastmed ja arenduse plaan. Seejärel luuakse ka algeline andmebaasi skeem, kus märgitakse ära kõik vaja lähevad tabelid ja nende omavahelised seosed.

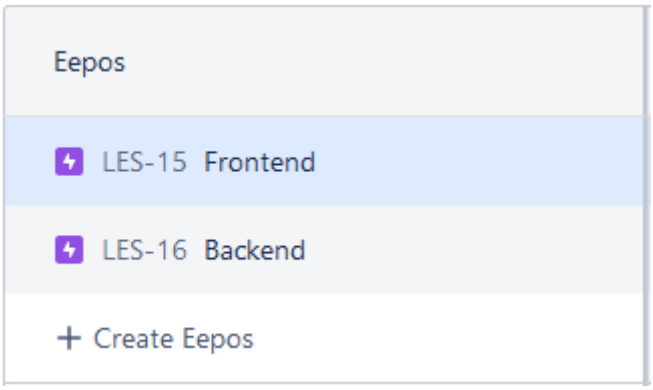
Teises pooles keskendutakse rohkem tehnilisele poolele, kus on põhjalikumalt juttu projekti graafilise liidese ja serveri arendamisest. Tuuakse välja koodinäited mingist veebirakendusel olevast funktsioonist ning samuti ka visuaalseid näidiseid rakendusest ning kasutajaliidesest.

3.1 Projekti ettevalmistus

Selles peatükis on keskendatud projekti ettevalmistusele. Põhjalikumalt on lahti räägitud Jira keskkonna kasutamine, toodud välja seal püstitatud probleemid, ülesanded ja eeposed, mis peavad olema täidetud antud rakenduse lahendamiseks. Kirjeldatud on ka andmebaasi skeemi, täpsemalt tabelite loomist, nende atribuute ja tüüpe ning nende omavahelisi seoseid.

3.1.1 Jira

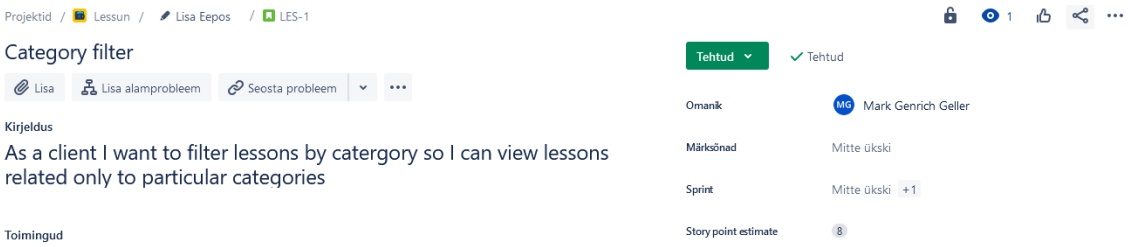
Käesoleva lõputöö rakenduse loomine sai alguse projekti loomises Jira keskkonnas. Järgmiseks etappiks oli luua vastavad eeposed ehk *epic*-ud. Antud rakendusele sai loodud vaid kaks *epic*-ut, üks kasutajaliidesele ja teine serveripoolsele liidesele.



Joonis 1. Projektile loodud eeposed.

Seejärel tuli luua probleemid antud eepostele ning lisada neile „Story point estimate“ ehk punktide hinnang, mis sümboliseeris antud probleemi või ülesande raskustaset või kui palju aega või panust on vaja iga probleemi või ülesande lahendamiseks. Üldjuhul kasutatakse eeltoodud punktide määramiseks Fibonacci arvu süsteemi, mitte tavapärast arvusüsteem. Põhjuseks sellele on raskuse arvestamise täpsus. Kui ühe ülesande raksustase on kolm ja teise neli, siis on nende keerukuse vahet palju raskem määrata ja sellest aru saada. Fibonacci korral on aga ühe punktitase kolm, siis sellest järgmine skaala järgi on viis ning siin on nende erinevus üksteisest märgatavam. [28] Ka tavapärasest tähtaja määramisest, mida samuti arenduses tihti kasutatakse, on see kasulikum, sest eemaldab ära sõltuvuse segavfaktoritest, mis otseselt pole projektiga seotud, näiteks nagu koosolekud, töömeilid ja intervjuud. Erinevalt tähtajalistest eesmärkidest premeerivad loo punktid arendajaid mitte kulutatud aja, vaid keeruliste ülesannete lahendamise põhjal, mis motiveerib ja juhib tähelepanu enne kvaliteedile ja väärtusele kui lahendamise kiirusele. Oluline on vaid määrata loo punktide alampiir ja ülempiir. [29]

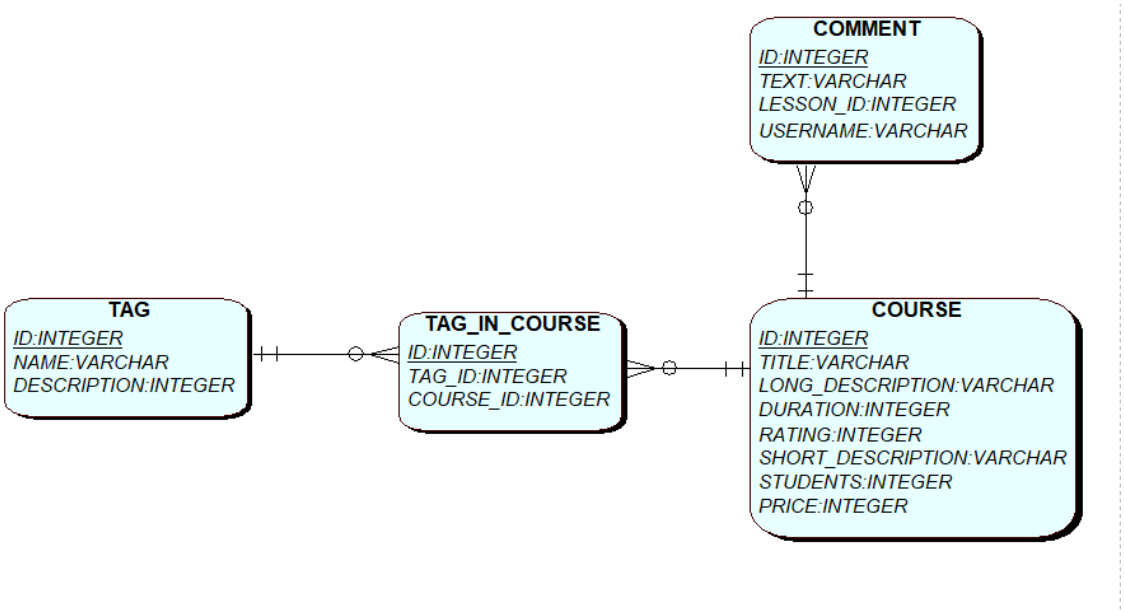
Igale kasutajaloole on määratud lisaks loo punktidele nimetus, mis üldjuhul peab väga lühidalt ära näitama, mis funktsionaalsust antud rakendus peab omama. Lisaks peab olema ka funktsionaalsuse põhjalikum kirjeldus, mille ülesehitus vastab üldjuhul mallile: „Kliendina soovin ma [tegevus, mida soovitakse teha] selleks, et [miks mainitud tegevust soovitakse teha].“ (Lisa 2)



Joonis 2. User Story täpne kirjeldus ja story point estimate.

3.1.2 Andmebaasi skeem

Järgmine etapp ettevalmistusel oli andmebaasi skeemi loomine ja selle visualiseerimine kas paberil või programmis. Skeemi loomine sai alguse tabelite loomisega ning atribuutide lisamisega ning nende tüübi määramisega. Kokku oli viis tabelit ning nendest üks oli seostabel. Suurim tabel antud süsteemis oli kursuste tabel, kuhu läksid kõik andmed kursustest. Atribuutideks oli eelkõige ID, mis oli määratud ka *primary key*-ks ehk tabelivõtmeks. Lisaks sellele olid tabelil ka sellised tunnused nagu pealkiri, lühikirjeldus, pikk kirjeldus, kursuse kestvus nädalates, õpilaste arv ehk kui mitmed inimesed on antud kursust läbinud, hind ja hinne. Samuti oli ka seosed teiste tabelitega. Üheks selliseks tabeliks oli tagasiside tabel, mis sümboliseeris kursuse individuaalset tagasisidet. Kursuse tabeliga või olla ühendatud null kuni mitu erinevat tagasiside tabelit, mis tähendab, et nende kahe tabeli vahel kehtis üks-mitmele ehk *one-to-many* seos. Kolmandaks tabeliks oli *Tag* tabel, mis sümboliseeris kursuste kategooriat. Selle tõttu, et nende kahe tabeli vahel kehti mitu-mitmele seos ehk *many-to-many* seos, pidi looma ka neljanda tabeli, milleks oli tavaline seosetabel, mis hoidis ennast andmeid, millised kategooria tabelid milliste kursuste tabelitega seotud on.



Joonis 3. Andmebaasi skeem põhitabelitega.

3.2 Projekti arendus

Selles peatükis keskendutakse rohkem veebirakenduse arendusprotsessile. Kirjutatakse lahti ning põhjendatakse, mida tehti nii graafilise kui ka serveri poolse liidese arendamise käigus.

3.2.1 Graafiline liides

Projekti koodi kirjutamine sai alguse *front-end*i valmistamisest. Esimese asjana otsiti välja sobiv veebilehe mall, mis antud lahendusele sobiks. Mainitud protseduuri eeliseks oli olemasoleva HTML ja CSS koodi ärakasutamine ning modifitseerimine enda nõuete järgi. Järgmisena oli valitud mall pandud valitud raamistiku, milleks eelnevalt mainitud analüüsi käigus selgunult oli React.js, ning jagada HTML kood algelisteks komponentideks, mis eemaldaks koodi ja teksti kordust ning võimaldaks erinevaid osasid taaskasutada. Sellele järgnes imiteeritud andmete loomine raamistiku sees, nende kasutamine ning korrapärase liidese loomine õigete komponentide ja teenustega, mis tagaks graafilise liidese korrapärase toimimise juba enne serveriga ühendamist.

React.js kasutab komponente, mis võimaldab jagada kasutajaliidese mitmeks erinevaks iseseisvaks tükkiks, mida on võimalik hiljem koodis taaskasutada. Seda võib vaadelda ka kui tavalist funktsiooni, millele juurde on üldjuhul lisatud ka HTML kood ning seetõttu tagastab mingisuguse kasutajaliidese osa. Sarnaselt tavalisele Javascript või Typescript funktsioonile on ka React.js komponendile võimalik anda kaasa parameetreid. React.js

komponente saab defineerida nii klassina kui ka eraldiseisva funktsioonina, mis osutust ka valikuks käesoleva veebirakenduse arendamise käigus. [30]

Sellele järgnes navigatsiooni loomine antud veebilehel ning komponentide omavaheline seostamine, mitte linkide vahel, nagu oli HTML-is sisse kirjutatud, vaid kasutusele oli võetud React.js enda poolt pakutatavad lahendused. Üks nendest, mida on kasutatud käesolevas rakenduses on React *Router library*. Sellega imporditakse rakendusse spetsiaalsed komponendid, mis muudavad liikumise lehtede vahel mugavamaks ja efektiivsemaks. Eeliseks tavalise HTML koodis oleva lingi ees on see, et komponentidele on võimalik anda kaasa ka andmed ja muutujad, mida saab järgmisel lehel edasi kasutada. Ka koodi väljanägemine muutub esteetilisemaks ning arusaadavamaks. [31]

```
const App = () => {
  return (
    <Router>
      <html lang="en">
        <HeadLinks />
        <body className="regular-navigation">
          <div id="master-wrapper">
            <Header />
            <Switch>
              <Route exact path="/"><Home /></Route>
              <Route exact path="/Home"><Home /></Route>
              <Route exact path="/Login"><Login /></Route>
              <Route exact path="/Register"><Register /></Route>
              <Route exact path="/AboutUs"><AboutUs /></Route>
              <Route exact path="/ContactUs"><ContactUs /></Route>
              <Route exact path="/NewsList"><NewsList /></Route>
              <Route exact path="/Article"><Article /></Route>
              <Route exact path="/Courses"><Courses /></Route>
              <Route exact path="/Courses/:id"><SingleCourse /></Route>
            </Switch>
            <Footer />
          </div>
          <FooterScriptLinks />
        </body>
      </html>
    </Router>
  )
}
```

Joonis 4. App.tsx kood, mis paneb kokku kõik rakenduse komponendid ning seob nad *Router*-iga kokku.

Lehtede vaheline navigatsioon on vaid võimalik juhul, kui rakenduse komponendid on *Switch* ja *Route* tagide vahele pakitud (Joonis 3), kuhu on ka lisatud nimetus ning URL-i osa. Kõik toimub App.tsx failis, kus on toodud välja kõik komponendid ning nendest on

pandud seal kokku ka terviklik veebirakendus ehk seda võib nimetada ka juhtkomponendiks. [31]

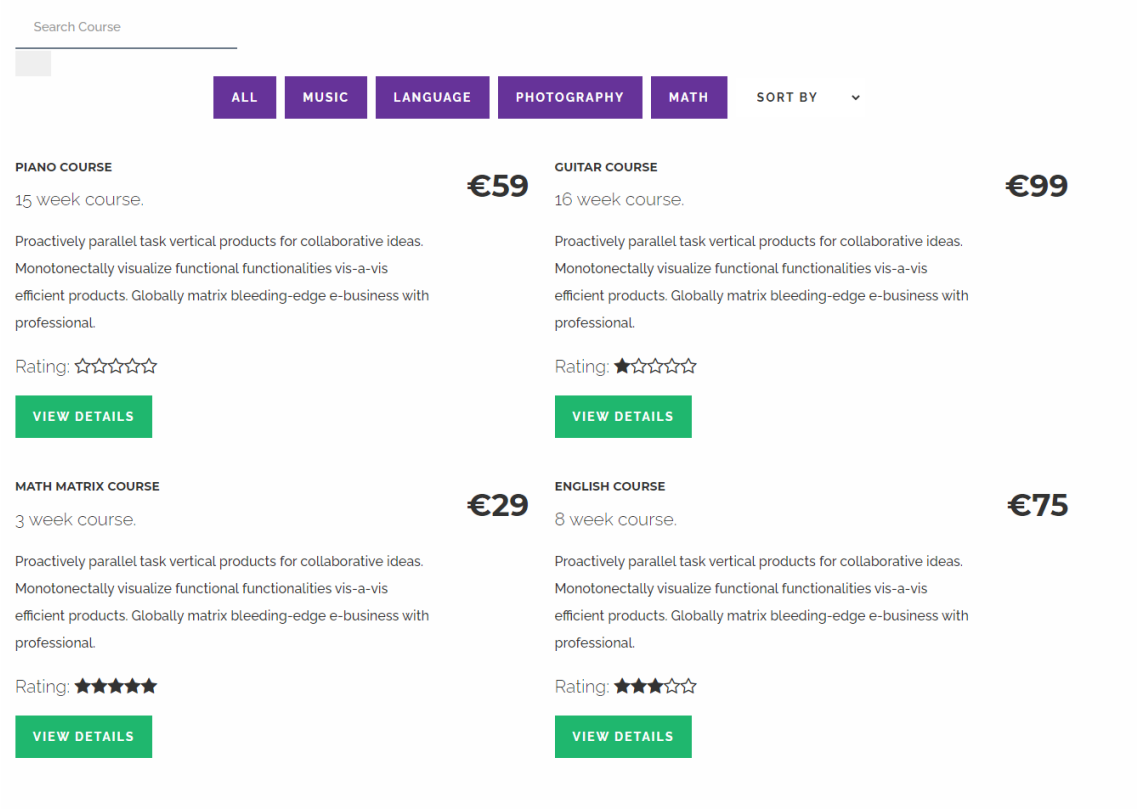
Joonis 5. Navigatsiooni lahtri näidis.

Oluline oli luua võimalikult täpsed ja hästi töötavad komponendid, et graafiline liides oleks töökorras juba ilma lisatud serverita. Selle jaoks olid loodud spetsiaalsed objektid ja list nendest objektidest, mis simuleeris andmete kättesaamist andmebaasist. Objektid olid loodud liidestena vastavalt ülaltoodud andmebaasi skeemile vastavate atribuutidega. Ka kasutajaliideses pidid tabelite järgi tehtud liidesed olema omavahel seotud, mis väljendus sellega, et kursusel oli nii *Tags* kui ka *Review* tüüpi järjend väljadena. Ülejäänud väljad olid string või number tüüpi (Joonis 5).

```
export interface ISingleCourse {
    id: string;
    title: string;
    duration: number;
    students: number;
    shortDescription: string;
    longDescription: string;
    price: number;
    rating: number;
    picturePath: string;
    tags: ITag[];
    reviews: IReview[];
}
```

Joonis 6. Kursuse liides selle atribuutide ja nende tüüpidega

Edasi hakkas arendus Jiras moodustatud kasutajaloode järgi. Alguses olid lahendusele võetud lihtsamad ning samas olulisemad ülesanded, nagu näiteks kursuste kuvamine lehel. Igal kursusel pidi olema ka nähtav nendele vastav hinnang, kursuse pikkus nädalates, lühikirjeldus ja pealkiri. Samal lehel pidi olema ka sektsioon olemasolevate kategooriatega, mille järgi oli võimalik kursuseid filtreerida, et kitsendada valik, vaid sellele valdkonnale, mis kasutajat huvitab. Samuti oli võimalus ka välja selekteeritud kursuste sorteerimine kas hinnangu, pikkuse või hinna järgi. Lehel on olemas ka dünaamiline otsingumootor, mis võimaldab kasutajal otsida kursuseid pealkirja järgi (Joonis 6).



Joonis 7. Kursuste avaleht.

Kursuste kuvamisel oli kasutatud React.js poolt loodud *useEffect*() *hook*-i, mis võimaldab teha kõrval toiminguid komponentides. [32] Sama lahendus oli kasutatud ka sorteerimise puhul, mille tulemusel oli kursuste komponendis kaks *hook*-i ning oli oluline ka nende järjestus, kuvamise efekt pidi olema enne sorteerimise oma (Joonis 7-8).

```
useEffect(() => {
    const callApi = async () => {
        const data = await CourseApi.getAll();
        console.log('data', data);
        setCourses(data);
    };

    callApi();
}, [courses.length]);
```

Joonis 8. Kuvamiseks kasutatav *useEffect*() *hook*.

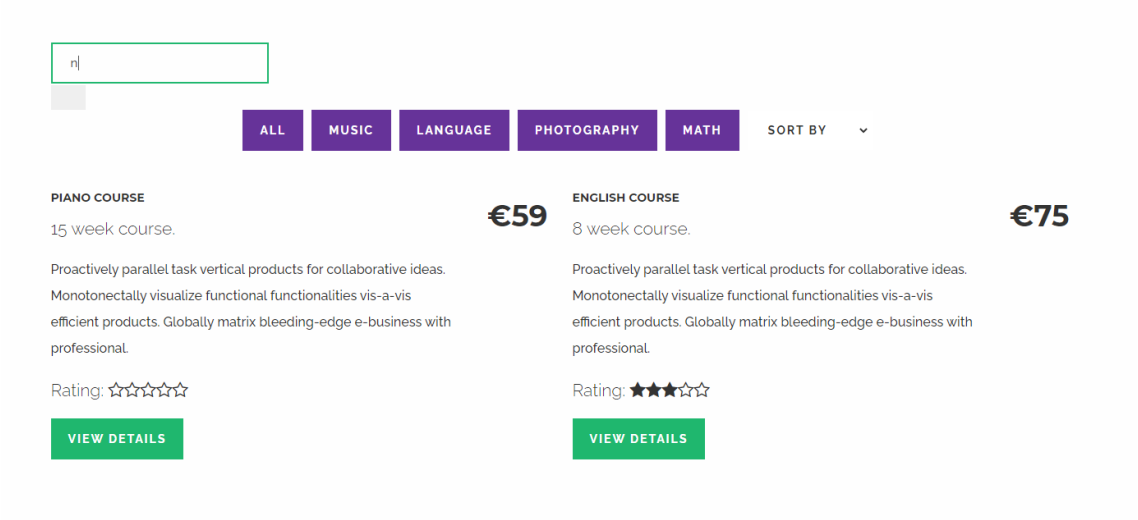
```
    useEffect(() => {

        const sortCourses = (type: string) => {
            const types: any = {
                price: 'price',
                rating: 'rating',
                duration: 'duration'
            };
            const sortProperty: string = types[type];
            const sorted = [...courses].sort((a: any, b: any) => a[sortProper
ty] - b[sortProperty]);
            setCourses(sorted);
        };

        sortCourses(sortField);
    }, [sortField])
```

Joonis 9. Sorteerimise *useEffect*() *hook*.

Igal React.js komponendil on olemas enda *state* ehk olek, kuhu saab salvestada erinevaid muutujaid ja andmeid. Selle muutumisel hakkab muutuma ka komponendi välimus ja informatsioon, mida läbi selle kuvatakse. [33] Kui kuvamisel ja sorteerimisel oli kasutatud eraldi *useEffect*() *hook*-i, läbi mille toimus suhtlemine komponendi olekuga ja seal olevate andmetega, siis nii otsingu kui ka filtreerimise puhul oli otsene *state* andmete kasutamine. Filtreerimisel oli igale lahtrile määratud olemasolevate kategooriate nimetused ning oli lisatud *onClick*() funktsionaalsus HTML koodi reale. Antud lisa tähendas seda, et mingisuguse lahtri vajutusel salvestatakse selle lahtri nimetus ning kuvatakse vaid need kursused, mille kategooria sisaldab valitud filtrit (Joonis 9). Otsimisega oli rakendatud sarnast põhimõtet, kuid oli lisatud ka eraldi funktsioon, mis jälgis otsingulahtri muutumist. Selle põhjuseks oli asjaolu, et rakendusele oli vajalik luua dünaamiline otsingumootor, mille jaoks ei ole vaja nuppu ning komponent kuvab automaatselt kõik vajalikud kursused tähehaaval välja. Selline otsingumootor on palju mugavam, kiirem ning ka efektiivsem. Erinevalt *onClick*() funktsioonist, tuli sellel juhul kasutada *onChange*() ning lisada juurde ka funktsioon *handleChange*() (Joonis 10), mis võttis parameetriks HTML *tag*-i elemendi ning funktsiooni sees lisati otsingus tulnud sisu komponendi olekusse, mille järgi otsiti kursuseid, mille nimi vastaks *state*-is oleva sisuga.

```
const handleChange = (target: EventTarget & HTMLInputElement) => {
        if (target.type === 'text') {
            setSearchField(target.value);
        }
}
```

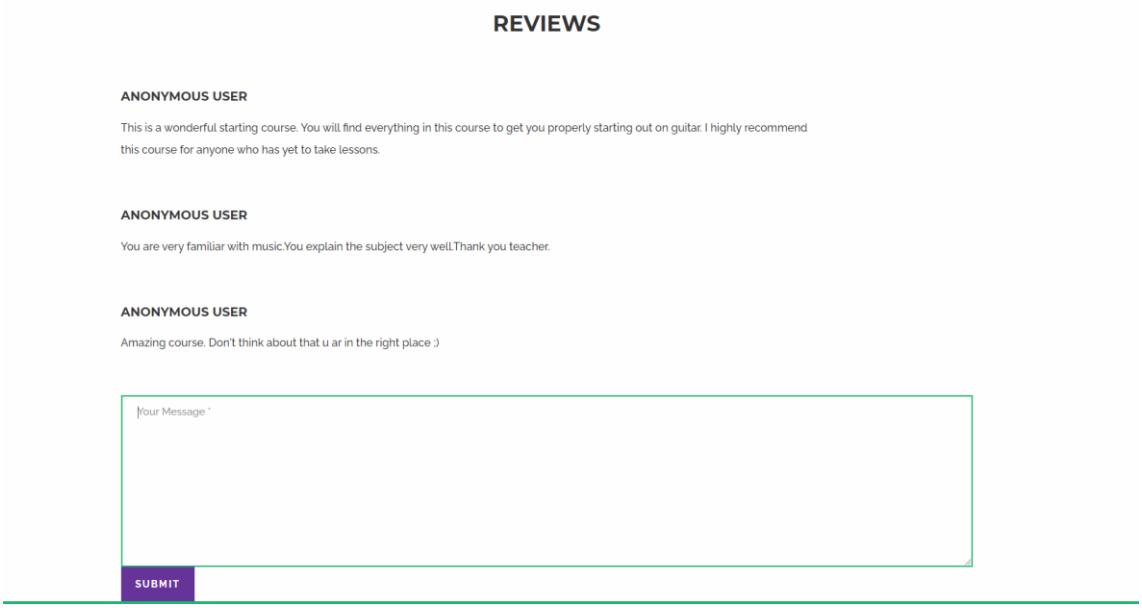
Joonis 10. React *handleChange*() funktsioon, mis salvestab otsingulahtrisse sisestatud andmed.



Joonis 11. Kõik kursused pärast otsingulahtisse ühe tähe sisestamist.

Samal kursuste avalehel on kasutajatel, kes on õpetajate rollis, võimalus panna üles ka enda loodud kursuse. Kursuse moodustamisel, peab autor lisama pealkirja ning vajalikud kursuse kirjeldused ja ülevaate. Samuti lisama kursuse pikkuse, lingi videole ning võimalus on sisestada ka pilt, mida kursusel kuvatakse. Kui kõik vajalikud väljad ja andmed on täidetud saadetakse *Post* päring serverile, mis lisab läbi mitme kihi vajalikud andmed baasi ning loob uue kursuse veebilehele. Pikemas perspektiivis oleks plaanis lisada rakendusele vahekontroll, mis kursuse lisamise järel saadaks informatsiooni alguses tiimile, kes selle üle vaataksid. Alles kui kursus on nende poolt kinnitatud ja luba antud, siis lisatakse see andmebaasi läbi eelmainitud *post* päringu ja serveri kihtide.

Iga kursuse all on roheline nupp, millele vajutades viib eelmainitud React.js Router vastavalt kursuse ID-le kursuse lehele, kus on lahti seletatud antud kursuse informatsioon pika kirjelduse näol. Samuti on välja toodud ka tagasiside, mida võib iga kursuse alla kirjutada. Toodud on välja ka teised atribuudid nagu kursuse hind, pikkus, kui mitu õpilast on antud lisatundi kasutanud.



Joonis 12. Tagasiside sektsioon kursuse lehel.

Kui komponendid olid valmis ning rakenduse graafiline pool töötas vigadeta ning oli visuaalselt korrektne ja selge, alustati ühendamist serveriga. Selle jaoks oli loodud eraldi klass, mis vastutas selle eest. Klassi eesmärgiks oli saata serverile kasutaja poolt küsitud päringuid, mis seejärel pidid tegema sellele vastava toimingu. Kasutusele oli võetud Axios, mis on üks populaarsemaid HTTP-kliente (*Hypertext Transfer Protocol*) nii brauserite kui ka Javascript raamistike seas. Põhjuseks selle lihtne ja arusaadav kasutus ning samuti on hõlpsasti kohandatav enda soovide järgi. Lisaks töötab see ka suurepäraselt nii React.js kui ka paljude teiste populaarsemate raamistikega. Axios pakub head kaitset saitidevaheliste taotluste võltsime eest kui ka tugiteenust päringute ja JSON-i automaatseks teisendamiseks. [34]

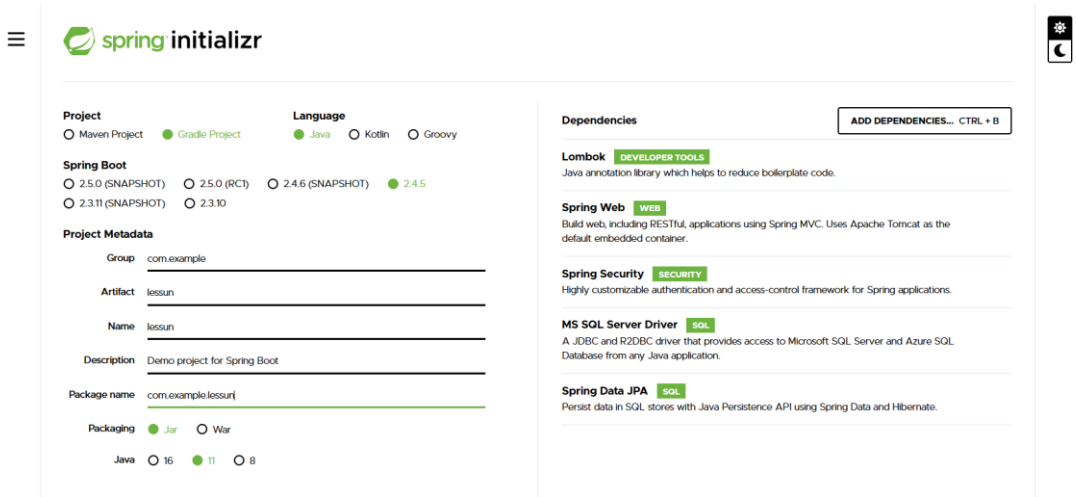
Nii kasutajaliides kui ka server vajavad ülalhoidu ning igapäevast kontrolli ja täiendamist. Mis puudutab klindi vajadusi ja nõudeid, siis veebirakenduses on loodud selle jaoks eraldi leht, kus kasutaja täidab vormi kirjeldades täpselt probleemi, mis tal esineb või ettepanekuid, kuidas antud rakendust saaks täiendada, kas funktsionaalsuse poolest või välimust. Vormi täitmisel oleks teade saadetud arendajate ühisele meilile, kus probleemi uuritakse ning otsustatakse, kas antud lahendus oleks tegelikkuses piisavalt efektiivne ning selle põhjal tehakse ka otsus. Üldiselt uue välimuse või kasutajaliidese funktsionaalsuse parandamine või lisamine ei ole React.js raamistiku puhul nii raske. Seda tänu komponentidele, mis võimaldavad igat graafilise liidese elementi vaadelda ja parandada eraldi, mitte segades sellega teiste komponentide toimimist.

3.2.2 Serveripoolne liides

Veebirakendus oli loodud REST API arhitektuuril, mis üldjuhul tähendab, et kehtib klient-server mudel. Kasutajaliidese arendamine ja ülespanemine oli eelnevalt lahti räägitud, kuid selles peatükis tuleb täpsem kirjeldus serveri arendamisest ning selle ühendamisest graafilise liidesega.

Programmeerimiskeeleks, serveri arenduseks, oli valitud juba eelnevates peatükkides mainitud Java, täpsemalt Java Spring raamistik. Tegu on ühe kõige populaarsema rakenduste arendamises kasutatava raamistikuga. Üle miljoni arendaja kasutab Spring raamistiku suure jõudluse, kergesti testitava ja korduskasutatava koodi kirjutamiseks. Usaldusväärseks arendajate seas muudab asjaolu, et tegu on avatud lähtekoodiga , mis võimaldab igal ühel sellele ligi pääseda ning muuta enda soovide järgi. Kõige enam luuakse Springis veebirakendusi sellepärast, et see sisaldab endas ka mitmeid laiendusi, mis muudavad arenduse veelgi mugavaks ja efektiivsemaks. [35]

Serveri arendus sai alguse Spring Initializer veebilehelt [36], mis teeb kasutajale valmis Spring raamistikul projekt (Joonis 12). Rakenduse loomiseks on vaja sisestada nii projekti nimi, pakki nimi, kus rakendus peaks asuma ja lühikirjeldus, mis üldjuhul läheb README faili. Lisaks sellele on võimalik valida ka projekti, kuhu sisse rakendus on pakitud, Springi versioon, pakki failitüüp, Java versioon ning milles projekti arendus hakkab olema. Kõige lõpetuseks on võimalik kohe alguses lisada ka laiendused, mida läheb arendusel kindlasti vaja. Seejärel tehakse valmis .zip fail, mille sees asub loodud Spring projekt.



Joonis 13. Käesoleva projekti andmed Spring Initializeris (eeldades, et projekti nimi on „Lessun“).

Projekti allalaadimisel ja lahti pakkimisel luuakse esimese andmebaasi tabelite mudelid nagu oli tehtud ka graafilise liidese korral (Joonis). Erinevalt kasutajaliidesest on Javas arvude tüüpe palju rohkem, näiteks lisaks tavalisele täisarvulisele Integerile on olemas ka Long ja Double tüübid. Käesolevas veebirakenduses oli ID jaoks võetud Long tüüp, sest lubab hoida endas palju suuremaid arve kui näiteks Integer, seega andmeid andmebaasi oleks võimalik panna palju rohkem. Seevastu kursuse kestvuse tüübiks piisas Integer-ist, sest tegu on mitte suurte täisarvudega, mis tõttu nii Long kui ka Double oleksid antud väljade jaoks asjatud. oleksid Pikemas perspektiivis oleks plaanis kasutusele võtta GUID (*Globally Unique Identifier*) ID määramiseks. GUID või ka UUID (*Universally Unique Identifier*), on universaalsed unikaalsed identifikaatorid, mis on ligi 128 bitised viitenumbrid, mida üldjuhul genereerimisel ei kordu, mistõttu suurtemate ja keerukamate süsteemide puhul, kus andmete kogus on vastav, on soovitatav kasutada just selliseid unikaalseid identifikaatoreid ID määramisel. Kõik GUID variandid jälgivad ühte ja sama struktuuri xxxxxxxx-xxxx-Mxxx-Nxxx-xxxxxxxxxxxx, kus M tähistab versiooni ja N kõige olulisemad bitid tähistavad varianti ning x kohtadele lähevad üldjuhul numbrid ja tähed segamini. [37]

```
@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "courses")
public class Course {

    @Id
    @SequenceGenerator(name = "my_seq", sequenceName = "seq1", allocationSize
= 1)
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "my_seq")
    private Long id;
    private String tag;
    private String title;
    private Integer duration;
    private Integer students;

    @Column(name = "short_description")
    private String shortDescription;

    @Column(name = "long_description")
    private String longDescription;

    private Long price;
    private Integer rating;

    @Column(name = "picture_path")
    private String picturePath;

    @ElementCollection
    @CollectionTable(name = "tags", joinColumns = @JoinColumn(name =
"course_id", referencedColumnName = "id"))
    private List<Tag> tags = new ArrayList<>();

    @ElementCollection
    @CollectionTable(name = "reviews", joinColumns = @JoinColumn(name =
"course_id", referencedColumnName = "id"))
    private List<Review> reviews = new ArrayList<>();


}
```

Joonis 14. Kursuste tabeli mudel serveris.

Paljudel objektidel ja klassidel, kaasaarvatud kursuste mudelil nagu oli näha Joonisel 13 oli kasutatud annotatsioone. Annotatsioonid on üldjuhul mõeldud selleks, et anda põhjalikuma ülevaate klassist, liidesest või funktsioonist, millele see on määratud. Enamus ajast ei oma annotatsioon erilist funktsionaalsust, algab „@“ sümboliga ning peamiselt on mõeldud arendaja ja kompilaatori jaoks. [38] Lisaks Java sisseehitatud annotatsioonidele, kasutavad seda lahendust ka mitmed sõltuvused, näiteks nagu Lombok. Tegu on Java laiendusega, mis mõjutab programmi vaid kompileerimise tasemel. Lomboki eesmärk on vähendada koodi kirjutamist ning sellega kaasnevaid vigu. Ühtlasi parandab see ka koodi loetavust. Näiteks, selle asemel, et kirjutada lähtekoodi

välja kõik väljade *setter*-id, *getter*-id, *equals*, *hashcode* ja *toString* meetodid, piisab vaid Lomboki @Data annotatsioonist klassi kohal nagu on näha ka Joonisel 13. Kompileerimise ajal lisab Lombok ise kõik kirjutamata meetodid „.class“ faili ning tänu sellele kogu funktsionaalsus jääb püsima. [39]

Valdav osa tänapäeva rakendusi väldib serveris otsest suhtlemist andmebaasiga ja informatsiooni vaheldamist otse tabeli mudelitega. Selle asemel kasutatakse DTO-si (*Data transfer object*), mis on üldjuhul andmebaasi tabelite koopiad, kuid nendes puudub igasugune funktsionaalsus ning on peamiselt mõeldud selleks, et edastada andmeid kasutajaliidese, serveri ja andmebaasi vahel. Eriti keerulisti süsteemide ja komplitseeritud tabelite puhul, mis sisaldavad palju erinevaid välju, kasutatakse DTO lahendust ka selleks, et osa tabeli sisust kasutajate eest ära peita, mis omakorda viitab rakenduse üldisele turvalisusele. Kuna käesoleva projekti puhul on tegu üpris kerge ning tavalise struktuuriga, siis DTO ülesehitus oli peaaegu samasugune nagu ka andmebaasi tabelite mudelitel. [40]

Nagu eelnevalt sai mainitud, andmebaasile ligipääsemine ei tohi olla otsene, vaid peab sisaldama vahekihte. Esiteks võimaldab see parema rakenduse testimise ning teiseks, eemaldab ka ebavajaliku koodi dubleerimise, mis oluliselt kergendab selle haldamist. Üheks võimaluseks vahekihti tekitamiseks oli luua rakendusele eraldi *repository* liides ja *service* klass, mis sisaldasid endas vajaliku loogikat ja funktsionaalsust andmebaasiga suhtlemiseks ja andmete käsitlemiseks. Spring raamistik võimaldas koodi kirjutamist veelgi vähendada, pakkudes JPA *repository* liidest, mis omab endas kõik vajalikud meetodid ja kogu vajaliku funktsionaalsuse selle taga. Serveris oli vaja luua lisameetodid, mis kasutaksid *repository* meetodeid vastavalt andmebaasi mudelitele. [41]

Viimane osa serverist hõlmas controlleri loomist, mis oleks otsene suhtlusvahend kasutajaliidesega, mis võtab esimesena vastu kasutaja poolt saadetud päringud ning rakendab vastavalt sellele õiget meetodit. Iga päringu kohta loodud funktsioon kutsub omakorda vastava *service* klassis loodud meetodi, mis läbi *repository* etappi teeb vajalikud toimingud andmebaasis ning tagastab tulemuse. [42]

Kasutajaliidese puhul oli räägitud ülalhoidmisest ning sellega seonduvatest detailidest. Ka serveri puhul oleks ülalhoid ning funktsionaalsuse muutmine vastavalt kasutaja soovidele ja tehnoloogia arengule vajalik. Kõik mis puudutab kliendi enda poolt pakutud

mugavuste lisamisega või probleemidega, oli lahti räägitud eelmises alampeatükis. Kasutaja täidab vormi, kus kirjeldab täpselt probleemi, lisatoiminguid, mida rakendus võiks teha või välimuse muutmise eelistusi. Seejärel saadetakse vorm automaatselt ära, kus seda põhjalikult uuritakse ning tehakse ära vajalikud toimingud. Tarkvara uuenemise puhuks oli antud serveri programmeerimiskeeleks valitud Java. Põhjuseks sellele, mida oli mainitud ka valiku tegemisel, Java on hästi paindlik keel, mis koodi kirjutamise ja kompileerimise hetkest võib töötada erinevatel platvormidel täpselt samamoodi, ilma mingisuguse raskuseta. Samuti ei tekitaks ka tarkavara uuendused mitte mingit segadust, sest kõik Java vanemad versioonid jooksevad sama hästi ka uuematel versioonidel, mistõttu koodi parandamine ei oleks kohustuslik ja juhul kui oleks, siis oleks seda väga kiire ja kerge teha.

3.3 Tulemus

Käesoleva projekti arendamise tulemusel sai valmis korrapärane REST API veebirakendus töötava kasutajaliidese ja serveriga, mis omakorda oli ühendatud andmebaasiga. Kursuste lisamisel saadab graafiline liides serverile Post päringu, mis läbi controlleri, *service* ja *repository* kihtide saadab toimingu andmebaasile, kus lisatakse valitud kursus andmebaasi ning tagastatakse loodud kursuse objekt, mis kasutajaliidesesse jõuab JSON-ina. Kursuste kuvamisel saadab kasutajaliides *Get* päringud, mille tulemus tagastab server kas järjendi kõikidest andmebaasis olevatest kursustest või vastava ID-ga kursuse, mis liigub DTO-na ning kasutajaliides tagastab mainitud objekti samuti JSON-ni, renderdades objekt liidesesse ning kuvades korrapäraselt kasutajale ekraanil. Lisaks sellele saab kasutaja lisada valitud kursusele ka tagasiside, mis annab ka tulevastele kasutajatele kui ka veebilehe ülalpidajatele ülevaate antud kursusest, selle ülesehitusest ja temaatikast ning selle täpsusest. Oma soovidele vastavalt on kasutajal võimalik kursuseid ka filtreerida kategooria järgi, sorteerida pikkuse, hinnangu ja hinna järgi ning otsida nime järgi.

Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua funktsioneeriv veebikursuseid pakkuv veebirakendus, mis sisaldaks endas erinevaid lisakursuseid erinevatest valdkondadest. Veebilehe kasutajateks oleksid õpilased, tudengid või teised inimesed, kes vajavad abi mingil kindlal teemal kindlas valdkonnas või lihtsalt soovivad enda teadmisi ja oskusi suurendada ning tööturul tugevamat konkurentsi pakkuda. Teiseks kasutaja pooleks oleksid nii-öelda õpetajad, kes jagaksid oma teadmisi teistega ning aitaksid sellega ka teisi ning ühtlasi saaksid ka ise sellest kogemust ja tagasiside nende õpetamistele.

Rakenduse vahendite valikul viidi läbi põhjalik analüüs mitme alternatiivse võimaluse vahel. Valiku tegemisel oli peamiselt keskendatud vahendi aktuaalsusele ehk kui populaarne ja nõutud on kasutamise kogemus tööturul, selle sobivusele antud eesmärk ära täita ning ka olemasolevale kogemusele, mis tähendas, kui palju aega oleks töö autor pidanud mingi vahendi õppimisele panustama. Projektijuhtimise tarkvara valikuks osutus Jira, peamiselt selle populaarsuse tõttu. Serveri programmeerimiskeele valikul oli omavahel võrreldud viite erinevat C süntaksi keelt, millest võitjana väljus Java ning sellele loodud Spring raamistik. Põhjuseks oli selle aktuaalsus nii Maailma kui ka Eesti tööturul kui ka teatud eelised lahenduse ülesehituses, mis võimaldavad serveri ülalhoidmise muuta kergemaks ja efektiivsemaks. Kasutajaliidese puhul oli keeleks eelistatud võtta Typescript, oma lisade ning tüüpide olemasolu tõttu, mis vähendab vigade tekkimist ja müütab koodi lugemise arusaadavamaks. Raamistike seast osutus valikuks React.js oma lihtsuse, populaarsuse kui ka selle projektiga sobivuse tõttu. Ka komponentide kasutus eeltoodud raamistikus oli üks põhilisi punkte, mis tähendas, et graafilise liidese elemente saab käsitleda eraldiseisvatena ning koodi on võimalik ka palju lihtsamini taaskasutada.

Rakendus on loodud hajusa süsteemina kasutades klient-server mudelit. Serveri poolne liides on loodud kasutades kihilist struktuuri, mis aitab koodi paremini hallata ning muudatusi sisse viia. Lisaks sellele püsib sellisel juhul äriloogika muust koodist eraldi. Kasutajaliidese ja serveri vaheliseks andmete edastamiseks olid loodud ka eraldi DTO-d vastavalt andmebaasi tabelitele. Graafiline liides on loodud, et kuvada serveri poolt

tagastatud JSON objektid kasutajale arusaadaval moel. Kasutajal on võimalik kuvada kõiki kursuseid, üksikut kursust, lisada kursus juurde ja lisada tagasiside nii kursuste kohta kui rakenduse üleüldise funktsionaalsuse ja välimuse kohta. Lisaks sellele on kuvatud kursuseid võimalik ka filtreerida valitud kategooria järgi. Kuvatud abitunde on võimalik ka sorteerida hinna, pikkuse või hinnangu järgi ning otsida pealkirja järgi, mis kitsendab valiku tegemist veelgi.

Loodud veebikursuste rakendus lahendab püstitatud eesmärgi ja ülesanded. Samuti on ka edasi arendamise võimalusi, mis viiksid lahenduse teistest alternatiivsetest lahendustest veelgi rohkem ette. Peamine oleks muidugi andmebaasi ja serveri suurendamine, mis lubaks hoida suurt hulka andmeid. Samuti ka koostöö erinevate organisatsioonide ja koolidega, mis tagaks rohkem kliente ning suurendaks kasutatavust. Pikemas perspektiivis oleks oluline ka sertifitseerimis, mis võimaldaks kasutajatel saada kursuste läbimise eest tunnustatud sertifikaate, mida kasutajatel oleks võimalik näidata ette organisatsioonidele.

Kasutatud kirjandus

[1] „Pew Research Center, “ [Online]. Available: https://www.pewresearch.org/fact-tank/2017/05/16/todays-young-workers-are-more-likely-than-ever-to-have-a-bachelors-degree/. [Accessed: 10 April 2021].
[2] „CNBC Make It,“ [Online]. Available: https://www.cnbc.com/2018/08/16/15-companies-that-no-longer-require-employees-to-have-a-college-degree.html. [Accessed: 10 April 2021].
[3] „IBM,“ [Online]. Available: https://www.ibm.com/cloud/learn/rest-apis. [Accessed: 10 April 2021].
[4] „Udemy,“ [Online]. Available: https://about.udemy.com/. [Accessed: 10 April 2021].
[5] „Upskillwise,“ [Online]. Available: https://upskillwise.com/reviews/udemy/. [Accessed: 10 April 2021].
[6] „Coursera,“ [Online]. Available: https://about.coursera.org/. [Accessed: 10 April 2021].
[7] „Trust Radius,“ [Online]. Available: https://www.trustradius.com/products/coursera/reviews?qs=pros-and-cons. [Accessed: 10 April 2021].
[8] „Alexa,“ [Online]. Available: https://www.alexa.com/topsites.
[Accessed: 10 April 2021].
[9] „Barndon Gaille,“ [Online]. Available: https://brandongaille.com/11-pros-and-cons-of-youtube-in-education/. [Accessed: 10 April 2021].
[10] „Codeacademy,“ [Online]. Available: https://www.codecademy.com/about.
[Accesed: 10 April 2021].
[11] „Mentornaut,“ [Online]. Available: https://www.mentornaut.ee/about. [Accessed: 10 April 2021].
[12] „Atlassian,“ [Online]. Available: https://www.atlassian.com/software/jira/guides/use-cases/who-uses-jira#companies-and-teams-that-use-jira. [Accessed: 11 April 2021].

[13] „All that Saas,“ [Online]. Available: https://allthatsaas.com/roundup/best-jira-alternatives/. [Accessed: 11 April 2021].
[14] „Trello,“ [Online]. Available: https://trello.com/about. [Accessed: 11 April 2021].
[15] „ClickUp,“ [Online]. Available: https://clickup.com/blog/jira-alternative/. [Accessed: 11 April 2021].
[16] „Medium,“ [Online]. Available: https://medium.com/techloop/an-introduction-to-backend-development-and-rest-apis-b1a1a978821f. [Accessed: 11 April 2021].
[17] „Career Karma,“ [Online]. Available: https://careerkarma.com/blog/how-many-coding-languages-are-there/. [Accessed: 11 April 2021].
[18] „BitDegree,“ [Online]. Available: https://www.bitdegree.org/tutorials/most-used-programming-languages/. [Accessed: 11 April 2021].
[19] „Northeaster university degree programs,“ [Online]. Available: https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/. [Accessed: 11 April 2021].
[20] „Darwin recruitment, “ [Online]. Available: https://www.darwinrecruitment.com/blog/2019/03/future-php-dying-language. [Accessed: 11 April 2021].
[21] „Collonmade,“ [Online]. Available: https://www.collonmade.com/java-better-choice-for-web-application-development/. [Accessed: 11 April 2021].
[22] „TutorialsPoint,“ [Online]. Available: https://www.tutorialspoint.com/h2_database/h2_database_introduction.htm. [Accesed: 12 April 2021].
[23] „KeyCDN,“ [Online]. Available: https://www.keycdn.com/blog/popular-databases. [Accessed: 12 April 2021].
[24] „YoungWonks,“ [Online]. Available: https://www.youngwonks.com/blog/Introduction-to-Front-end-development. [Accesed: 12 April 2021].
[25] „TutorialsPoint,“ [Online]. Available: https://www.tutorialspoint.com/typescript/typescript_overview.htm. [Accesed: 12 April 2021].
[26] „VisualStudio Magazine,“ [Online]. Available: https://visualstudiomagazine.com/articles/2020/12/03/octoverse-2020.aspx. [Accessed: 12 April 2021].

[27] „Simform,“ [Online]. Available: https://www.simform.com/best-frontend-frameworks/. [Accessed: 12 April 2021].
[28] „RubyGarage,“ [Online]. Available: https://rubygarage.org/blog/how-to-estimate-with-story-points. [Accessed: 13 April 2021].
[29] „Atlassian,“ [Online]. Available: https://www.atlassian.com/agile/project-management/estimation. [Accessed: 13 April 2021].
[30] „React,“ [Online]. Available: https://reactjs.org/docs/components-and-props.html. [Accessed 14 April].
[31] „FreeCodeCamp,“ [Online]. Available: https://www.freecodecamp.org/news/a-complete-beginners-guide-to-react-router-include-router-hooks/. [Accessed 14 April].
[32] „Dmitri Pavlutin,“ [Online]. Available: https://dmitripavlutin.com/react-useeffect-explanation/. [Accessed 14 April 2021].
[33] „React,“ [Online]. Available: https://reactjs.org/docs/faq-state.html. [Accessed: 14 April 2021].
[34] „Design Revision,“ [Online]. Available: https://designrevision.com/react-axios/. [Accessed: 14 April 2021].
[35] „Tutorials Point,“ [Online]. Available: https://www.tutorialspoint.com/spring/spring_overview.htm. [Accessed: 15 April].
[36] „Spring Initializer,“ [Online]. Available: https://start.spring.io/. [Accessed: 15 April].
[37] „Guid,“ [Online]. Available: http://guid.one/guid. [Accessed: 15 April].
[38] „Utkarsh Jain,“ [Online]. Available: https://utkjain.medium.com/java-annotations-101-cd4d01e1470a. [Accessed: 15 April].
[39] „GeeksForGeeks,“ [Online]. Available: https://www.geeksforgeeks.org/introduction-to-project-lombok-in-java-and-how-to-get-started/. [Accessed: 15 April].
[40] „CodeProject,“ [Online]. Available: https://www.codeproject.com/Articles/1050468/Data-Transfer-Object-Design-Pattern-in-Csharp. [Accessed: 15 April].
[41] „Oracle,“ [Online]. Available: https://www.oracle.com/java/technologies/data-access-object.html. [Accessed: 15 April].
[42] „GeeksForGeeks,“ [Online]. Available: https://www.geeksforgeeks.org/front-controller-design-pattern/. [Accessed: 15 April].

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

Mina, Mark Genrich Geller

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Abikursuseid pakkuv veebirakendus“, mille juhendaja on Jaanus Pöial
   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.

17.05.2021

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.

Lisa 2 – Loodud kasutajaloode näidistabel

Kasutajaloo ID	Nimi	Kirjeldus
LES – 1	Kategooria filter	Kliendina soovin ma filtreerida kursuseid kategooria järgi selleks, et näha ainult valitud kategooria kursuseid.
LES – 2	Valida kursus	Kliendina soovin ma valida endale meelepärase kursuse selleks, et alustada õppimisega.
LES – 3	Otsida kursust	Kliendina soovin ma otsida kursust pealkirja järgi selleks, et leida otsitav kursus kiiremini.
LES – 4	Lisada kursus	Kliendina soovin ma lisada enda kursust veebilehele selleks, et õpetada teistele antud teemat.
LES – 5	Suur hulk erinevaid kursuseid	Teenusepakkujana soovin ma kuvada kasutajatele suurt hulka erinevaid kursuseid selleks, et kasutajatel oleks olemas võimalikult palju võimalusi.
LES – 6	Sorteerida kursuseid kestvuse järgi	Kliendina soovin ma sorteerida kursuseid kestvuse järgi, et valida endale sobivamad enne.
LES – 7	Kursuse hinnang	Kliendina soovin ma näha iga kursuse hinnangut selleks, et teada, kui usaldusväärne kursus on.

LES – 8	Kursuses osalejate arv	Kliendina soovin ma näha iga kursuse osalejate arvu selleks, et näha kui populaarne kursus on.
LES – 9	Kursuse tagasiside	Kliendina soovin ma näha iga kursuse tagasiside selleks, et teada, kui usaldusväärne kursus on lugedes mida teised on kirjutanud.
LES – 10	Jätta kursuse tagasiside	Kliendina soovin ma jätta iga kursuse alla enda tagasiside selleks, et kirjeldada enda kogemust kursusega.
LES – 11	Jätta veebirakenduse kohta tagasiside	Kliendina soovin ma jätta tagasiside veebirakenduse kohta selleks, et anda teenusepakkujale teada, mis on hästi ja mis halvasti.