TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

TalTech Virumaa kolledži publikatsioonide veebirakenduse loomine

Creating a web application for TalTech Virumaa College publications

RAKENDUSINFOTEHNOLOOGIA ÕPPEKAVA LÕPUTÖÖ

Üliõpilane: Airika Andruse
Üliõpilaskood: 131631
Juhendaja: Natalja Ivleva, lektor

Kohtla-Järve, 2021

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"...." ..... 20.....

Autor: .....
/ allkiri /

Töö vastab rakenduskõrgharidusõppe lõputööle/magistritööle esitatud nõuetele
"...." ..... 20.....

Juhendaja: .....
/ allkiri /

Kaitsmisele lubatud
"...." ..... 20.....

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS

Mina Airika Andruse (sünnikuupäev: 04.10.1991)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Virumaa Kolledži publikatsioonide veebirakenduse loomine, mille juhendaja on Natalja Ivleva,

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas 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 punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

TalTech Inseneriteaduskond Virumaa kolledž

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Airika Andruse 131631RDIR

Õppekava, peaeriala: RDIR02/12 - Rakendusinfotehnoloogia, tarkvara programmeerimine

Juhendaja(d): lektor Natalja Ivleva, natalja.ivleva@taltech.ee

Lõputöö teema:

(eesti keeles) TalTech Virumaa kolledži publikatsioonide veebirakenduse loomine

(inglise keeles) Creating a web application for TalTech Virumaa College publications

Lõputöö põhieesmärk:

1. Luua TalTech Virumaa kolledžile veebirakendus, kuhu salvestada kolledžist ilmunud publikatsioonid.

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Erinevate lahenduste uurimine, katsetamine	15.03
2.	Lõputöö dokumendi esimene osa – sisukord, sissejuhatus, sisuline osa (teooria, erinevad lahendused, nende kirjedamine, lahenduste realiseerimise meetotodid, võrdlus ja valik)	20.03
3.	Veebirakenduse koodi kirjutamine	20.02-20.03
4.	Lõputöö dokumendi teine osa – sisulise osa täiendamine (valiku täpsem kirjeldamine, praktilise osa kirjedamine)	17.04
5.	Veebirakenduse koodide kirjutamine ja testimine	23.03-10.05
6.	Lõputöö dokumendi viimane osa – sisulise osa lõpetamine (tulemused, järeldused, kokkuvõte)	16.05
7.	Lõputöö viimane kirjalik vormistamine	22.05

Töö keel: eesti **Lõputöö esitamise tähtaeg:** "28"mai 2021a

Üliõpilane: ..... "....."..... 20.....a
/allkiri/

Juhendaja: ..... "....."..... 20.....a
/allkiri/

Konsultant: ..... "....."..... 20.....a
/allkiri/

Programmijuht: ..... "....."..... 20.....a
/allkiri/

SISUKORD

EESSÕNA 7
LÜHENDITE JA TÄHISTE LOETELU 8
SISSEJUHATUS 9
1. TEHNOLOOGIATE VALIK 10
1.1 Veebirakenduse tehnoloogiad 10
1.1.1 PHP 10
1.1.2 Java 11
1.1.3 JavaScript ja Node.js 11
1.1.4 Veebitehnoloogia valiku põhjendus 11
1.2 Tagarakendus 12
1.2.1 Hapi 12
1.2.2 Koa 12
1.2.3 Express 13
1.2.4 Tagarakenduse valiku põhjendus 13
1.3 Eesrakendus 13
1.3.1 React 14
1.3.2 Vue 14
1.3.3 Angular 14
1.3.4 Eesrakenduse valiku põhjendus 14
1.4 Andmebaas 15
1.4.1 PostgreSQL 15
1.4.2 MySQL 16
1.4.3 MongoDB 16
1.4.4 Andmebaasi valiku põhjendus 16
1.5 Tehnoloogia valiku põhjendus 17
2. VEEBIRAKENDUSE LOOMINE 19
2.1 Veebirakenduse kirjeldus 19
2.2 Analüüs ja arendusnõuded 19
2.3 Loomisprotsess 21
2.4 Veebirakenduse struktuur 21
2.4.1 Projekti tagarakenduse struktuur 23
2.4.2 JWT Authentication 32
2.4.3 Projekti eesrakenduse struktuur 33
2.4.4 Veebirakenduse vaated 40

3. TULEMUS JA EDASIARENDAMINE 45
KOKKUVÕTE 47
SUMMARY 48
KASUTATUD ALLIKAD 49
LISAD 52

EESSÕNA

Käesolev lõputöö on valminud TalTech Virumaa kolledži tellimusena. Teema on valitud lähtudes organisatsiooni publikatsioonide rohkuse ja sobiva salvestamissüsteemi puuduse tõttu. Töö on kolledžiga seonduvate isikute või otseselt organistatsiooni puudutavate väljaannete süstematiseerimise, korrastamise ja talletamise tõhusama veebisüsteemi välja töötamine ja loomine. Lõputöö teema pakkus välja TalTech Virumaa kolledži turundus- ja kommunikatsioonijuht.

Lõputöö valmimisele on kaasa aidanud kontaktisikuna Annely Oone ja juhendajana telemaatika ja arukate süsteemide lektor Natalja Ivleva. Suur tänu toetavatele isikutele abi ja õpetuste eest.

Võtmesõnad: publikatsioon, veebirakendus, andmebaas, rakenduskõrgharidusõppe lõputöö

LÜHENDITE JA TÄHISTE LOETELU

CSS *Cascading Style Sheets,* märgistuskeel veebilehe paigutuse ja disaini loomiseks

DOM Eeskiri, objektide esitlemiseks veebilehel

GitHub Arendusplatvorm koodide, projektide, tarkvara versioonihalduseks ja koostööks

HTML *Hypettext Markup Language*, hüperteksti märgistuskeel veebilehtede loomiseks

HTTP *Hypettext Transfer Protocol*, hüperteksti edastusprotokoll

JavaScript Objektorienteeritud programmeerimiskeel

JSON *JavaScript Object Notation*, JavaScripti programmeerimiskeele alamhulgal põhinev andmevahetusvorming

JWT *JSON Web Token*, autentimise lubade avatud standard, mis määratleb viisi teabe turvaliseks edastamiseks osapoolte vahel JSON objektidena.

MongoDB andmebaasisüsteem

Mongoose MongoDB objektide modeleerimise tööriist

MVC *Model-View-Controller,* veebirakenduse arhitektuur (mudel, vaade, kontroller)

Node.js JavaScripti käivitamise keskkond

URL *Uniform Resource Locator*, teenust, asutust vm ressurssi tuvastav aadress internetis

SISSEJUHATUS

Publikatsioon (ingl k *publication*) on türkisena, internetis või muul viisil avaldatud tekst, näiteks artikkel, raamat. [1]

Veebirakendus (ingl k *web application*) on terviklik programm internetis, mille abil kasutaja täidab teatud ülesandeid. [2]

Veebirakendus on olemuselt hajus rakendusprogramm, mis ei sõltu platvormist, on klient-server-arhitektuuriga, kus kliendiks on veebibrauser ja serveriks veebiserver. Andmeid hoitakse ja töödeldakse peamiselt serveril ning andmevahetus toimub võrgu kaudu. [3]

Organisatsioonid, kes publikatsioone väljastavad, omavad erinevaid infosüsteeme, rakendusi või andmebaase, kus väljaandeid hoiustatakse. Päevakohased artiklid ilmuvad erinevates ajalehtedes või asutuste portaalides, mis on internetis lihtsasti leitavad ja kättesaadavad aga millest osa sisu on tasulised ja ajapikku lingid ehk URL-d aeguvad ning need ei ole enam ligipääsetavad.

Tallinna Tehnikaülikoolis (TalTech-is) on kasutusel sellised veebirakendused nagu STATION (station.ee) ja Eesti Teadusinfosüsteem ehk ETIS (etis.ee) ning Digikogu (digikogu.taltech.ee). Antud veebipõhised infosüsteemid hõlmavad kogu TalTech kui asutuse infovooge, lõputöid või neid tutvustavaid kokkuvõtteid ning muid teadusartikleid. Nendesse veebisüsteemidesse on kaasatud ka TalTech-i allasutuste publikatsioonid, mis tõttu on informatsiooni väga palju või mis sisaldavad eelkõige teaduslikke artikleid. Probleemi olemus seisneb selles, et TalTech Virumaa kolledžil puudub asutusesisene veebipõhine rakendus, kuhu koguda kõiki valdkondi hõlmav publikatsioonide kogumik.

Lõputöö eesmärk on luua TalTech Virumaa kolledžile veebirakendus nimetusega VikoPub, kuhu salvestada kolledžist ilmunud publikatsioonid. See platvorm ei asenda kolledži kasutuses olemasolevaid publikatsioonide infosüsteeme, mis erinevatel põhjustel ei ole sobinud, vaid hakkab paralleelselt eksisteerima.

Lõputöö ülesanneteks on järgnevad tegevused:

1. uurida sarnaseid lahendusi, programme ja tehnoloogiaid ning neid omavahel võrrelda;
2. teha REST API veebiteenused;
3. kasutada MongoDB andmebaasi;
4. luua JWT autentimine;
5. luua eesrakendus ja UI - kasutajaliides.

1. TEHNOLOOGIATE VALIK

Selles peatükis tutvustatakse erinevaid tehnoloogiaid, realiseerimise meetodeid, võrreldakse vahendite positiivseid ja negatiivseid külgi ning tuuakse välja rakenduse loomiseks sobiv valik.

Veebirakendus sisaldab kahte osa taga- ja eesosa ehk taga- ja eesrakendust ning andmeid hoitakse andmebaasis, mis peab ühilduma tagaosaga. Tagarakendus (ingl k *backend* või *back-end*) ehk serveripoolne osa on kasutajale nähtamatu, töötlev ja käitlev põhiosa programmist. Eesrakendus (ingl k *frontend* või *front-end*) ehk kliendipoolne osa, mis liidestab või ühendab kasutaja rakenduse tagaosaga.

Veebirakenduses kasutatakse REST API veebiteenuseid. REST API (ingl k *Representational State Transfer Application Programming Interface*) on veebiteenuste arhitektuuristiil ühtseks liidestamiseks kasutatav rakendusprogrammliides.

1.1 Veebirakenduse tehnoloogiad

Veebirakenduste loomiseks on mitmeid erinevaid võimalusi ja tehnoloogiaid. Teadatuntud programmeerimiskeeled nagu Python, Java, PHP, Ruby pakuvad erinevaid raamistikke veebilehekülgede tegemiseks.

Veebirakendus tehnoloogia valik sõltub eelkõige arendaja oskustest, tehnoloogia sobivusest ja tellija või kliendi nõudmistest. Analüüsides erinevaid võimalusi, tehnoloogiaid ja raamistikke, tuleb leida teenuse ja arendaja huvidele vastavalt parim lahendus.

Alljärgnevates alapeatükkides kirjeldatakse täpsemalt kolme veebitehnoloogiat või programmeerimiskeelt, PHP-d, Java-t ja JavaScripti koos Node.js-iga. PHP, Java ja JavaScript Node.js on valitud sellepärast, et neid tehnoloogiaid on autor õppinud, eelnevalt katsetanud ja kasutanud.

1.1.1 PHP

PHP (algselt ingl k *Personal Home Page* hilisemalt ingl k *Hypertetx Preprocessor*) on avatud lähtekoodiga üldotstarbeline skriptimiskeel. PHP on keskendunud peamiselt serveripoolsele skriptimisele, aga kasutatakse ka käsurea skriptimisel ja töölaua rakenduste loomiseks. [4]

PHP on lihtne. Seda on kerge õppida ja googeldades leiab palju materjale, koode, mida katsetada. Läbi parandamiste ja uuendamiste on PHP pidevalt arenev. PHP kood genereerib HTML-i ja see läbi on see platvormist sõltumatu. Eriti hea ühenduvus või suhtlus on PHP-l laienduse kaudu MySql andmebassiga. On olemas ka objektorienteeritud programmeeriimise (OOP) tugi, mis võimaldab suurendada funktsioonaalsust ja arendada modulaarsust. [5]

PHP raamistikest tuntumad on Laravel, Symfony, CodeIgniter, Zend, FuelPHP, Slim jne. Laravel on neist kõige populaarsem ja saavutanud märkimisväärse edu. Laravelil on võime hallata suuri ja keerulisi veebirakendusi. Slim vastupidi on just lihtsate ja väikeste rakenduste jaoks REST API raamistik, mis sisaldab rohkeid funktsioone. [6]

1.1.2 Java

Java on kompileeritav programmeerimikeel, mis kuulub Java Oracle Corporation omandisse. Javat kasutatakse erinevates mobiilirakendustes (eriti Androidi rakendustes), töölauarakendustes, veebirakendustes, veebi- ja rakendusserverites, mängudes, andmebaasi ühendustes jpm lahenduste loomisel. Java töötab erinevatel platvormidel, on üks populaarsemaid programmeerimiskeeli, seda on lihtne õppida ja kasutada, on kiire, turvaline ja võimas. Java on objektile orienteeritud keel, mis annab programmidele selge struktuuri ja võimaldab koodi taaskasutada, vähendades arenduskulusid. [7] Veebirakenduste arendamisel pakub Java Spring Boot-i, mis on sujuvam serveripoolsete HTML-rakenduste, REST API-de ja kahesuunaliste ning sündmustepõhiste veebisüsteemide väljatöötamiseks. [8]

1.1.3 JavaScript ja Node.js

JavaScript on tuntumaid veebiarenduse programmeerimiskeeli, millele on loodud palju raamistikke ja teeke ning mis võimaldab luua veebirakendusi nii serveri- kui ka kliendiosa pooltel ning lisaks veel mobiilirakendusi. JavaScriptist on välja arenenud uus versioon või teisisõnu arenduskeel TypeScript. TypeScripti eeliseks võib välja tuua selle, et see lisab koodi tüübiohutuse, mille käigus on lihtsam arengufaasis tabada JavaScriptiga seotud vigu ning muudab ka objektorienteeritud koodi arendamise JavaScripti jaoks lihtsamaks läbi sisseehitatud silumisvahendite kaudu. [9]

Node.js on avatud lähtekoodiga keskkond, mis kasutab asünkroonset programmeerimist ja võimaldab serveris JavaScripti käivitada. See ei ole programmeerimiskeel. Node.js on ehitatud Google Chrome V8 JavaScripti mootorile. Node.js välistab ootamise ja jätkab teiste töötlemise toimingutega veebisüsteemis ning käivitab üheahelalise, mitteblokeeriva, asünkroonse programmeerimise, mis on väga mälusäästlik. Node.js kasutab kliendipoolse kui ka serveri poolse osa jaoks JavaScripti. [10]

Node.js plussiks on see, et ta on mitteblokeeriv, haldab sisendit ja väljundit tõhusalt. Serveri haldamine ja eraldamine on sisse viidud, mis võimaldab uusi funktsioone sagedamini juurutada ja skaleerimist automatiseerida. [11]

1.1.4 Veebitehnoloogia valiku põhjendus

Veebitehnoloogiate PHP, Java ja Node.js vahel osutus loodava veebirakenduse valikuks JavaScript ja Node.js. JavaScripti ja Node.js kasutamist soovitati TalTech Virumaa

kolledži poolt ning oma erinevate võimaluste, uudsuse, autori huvi ja oskuste poolest eelistati samuti JavaScripti ja Node.js-i.

Stackoverflow 2020. aasta statistilised andmete põhjal programmeerimise, skriptimise ja märgistuskeelte seas on kaheksa viimast aastat olnud kõige populaarsemaks JavaScript, mida kasutatakse Node.js-is. Kategooria muud raamistikud ja tööriistad seas on kõige tuntumaks Node.js, mida pooled vastanutest on koodides kasutanud või kasutavad (Tabel 1.1). [12]

Tabel 1.1 Stackoverflow 2020. aasta analüüs (1,5,8 - asetsus Stackoverflow andmetes)

Programmeerimise, skriptimise ja märgistuskeelte populaarsus		Muude raamistike või tööriistade populaarsus (TOP3)	
JavaScript (1)	67,7%	Node.js	51,4%
Java (5)	40,2%	.NET	35,1%
PHP (8)	26,2%	.NET Core	26,7%
...		...	

JavaScript Node.js on mastaapne, suure jõudlusega, kiire ja kerge veebiraamistik, mis võimaldab samaaegselt käivitada mitu toimingut ning korduvkasutatavad ja kasutusvalmis komponendid säästavad aega. Node.js on võrreldes PHP ja Javaga palju kiirem ja tõhusam.

1.2 Tagarakendus

Veebirakenduse tehnoloogia valikust eelmises peatükis 1.1.4 lähtuvalt, milleks oli JavaScript ja Node.js vaadeldakse käesolevates alapeatükkides kolme tuntumat ja suuremat Node.js raamistiku Hapi-t, Koa-d ja Express-i. Hapi, Koa ja Express keskenduvad serveripoolsele ehk tagarakendusele ja mis kõik sisaldavad endas HTTP meetodit.

1.2.1 Hapi

Hapi või teise nimega ka Hapi.js (tuletatud Http-API-st) on avatud lähtekoodiga Node.js raamistik, mida kasutatakse veebirakenduste tegemiseks. Hapi-l on erinevaid unikaalseid funktsioone, mis võimaldavad arendajal ehitada turvalisi, võimsaid ja skaleeritavaid rakendusi. Koodi hügieen aitab arendajal kirjutada hallatavat, kontrollitavat ja leitavat koodi. Hapi blokeerib veateateid, mis võivad lekitada infot. Turvalisuse suurendamiseks kasutatakse krüpteeritud ja allkirjastatud küpsiseid, vahelduvaid või salajasi võtmete vaheldumist ja HTTP-turvameetodeid. [13]

1.2.2 Koa

Koa või Koa.js on Node.js veebiraamistik, mille on välja töötanud Expressi meeskond ja mille eesmärk on olla veebirakenduste ja API-de jaoks väiksem, väljendusrikkam ning

tugevam alus. Asünkroonimisfunktsioonide abil võimaldab Koa tagasihelistamisi tühistada ja tõsta oluliselt vigade käsitlemist. Koa ei komplekteeri oma tuumas ühtegi vahevara ja see pakub elegantset meetodite komplekti, mis muudab serveriosa kirjutamise kiireks ja nauditavaks. Vaatamata mõistlikult suure hulga kasulike meetodite pakkumisele on Koal väike ressursi kasutamine, kuna ühtegi vahevara pole komplektis. [14]

1.2.3 Express

Express või Express.js on minimaalne, paindlik ja lihtne Node.js veebirakenduste raamistik, mis on veebi- ja mobiilirakenduste arendamise toetamiseks. Express võimaldab HTTP päringute vastamise vahendamist, määratleb marsruutimist, mida kasutatakse HTTP-meetodi ja URL-i erinavateks tominguteks ning dünaamiliselt renderdab HTML-lehtede mallide ja argumentide põhjal. [15]

1.2.4 Tagarakenduse valiku põhjendus

Tagarakendus raamistike Hapi, Koa ja Express-i vahel osutus loodava veebirakenduse valikuks Express. Arendusplatvormi GitHub kohaselt on näha 2021 aasta aprilli seis Hapi, Koa ja Expressi populaarsusest (Tabel 1.2) [16] [17] [18]. Täpsemalt terminitest: positiivset tagasisidet jätnud kasutajate arv on näha tähtedena (ingl k *star*), kaasautorid on kasutajad, kes projekti koostööna arendavad (ingl k *contributors*) ja projekti kopeerimine on projekti dubleerimine ilma, et see algset projekti mõjutaks (ingl k *fork)*.

Tabel 1.2 GitHub-i populaarsus

Tagarakenduse raamistikud	Tähti	Kaasautorit	Projekti kopeerimist
Backend framework	*star*	*contributors*	*fork*
Hapi	13200	209	1300
Koa	31000	219	3000
Expresss	52700	262	8900

GitHubi järgi on Express antud kolmest raamistikust kõige populaarsem ja see on neist ka kõige vanem. Hea funktsionaalsuse tõttu on Express veebisüsteemide loomiseks sobilik raamistik, jättes autorile valikuvõimaluse rakendatavate tehnoloogiate ja arhitektuuri osas.

1.3 Eesrakendus

Järgnevates alapeatükkides tuuakse välja kolm populaarsemat JavaScripti raamistikku React, Vue ja Angular, mis on loodud veebirakenduse eesrakenduse loomiseks.

1.3.1 React

React või ka ReactJs on Facebooki loodud JavaScripti teek korduvkasutavate kasutajaliidese komponentide loomiseks. MVC (ingl k *Model-View-Controller*) arhidektuuris on React vaate osa ehk esindab tähist V. Andmevoog toimib ühesuunalisena. Reacti plussiks võib välja tuua, et see kasutab virtuaalset DOM-i (ingl k *virtual DOM* (*Document Object Model*)), mis on JavaScripti objekt ning see on kiirem kui tavaline DOM. Lisaks saab Reacti kasutada nii kliendi kui ka serveri ja teiste raamistike puhul. React rakendab ühesuunalist reaktiivset andmevoogu, mis vähendab ressurssi ja seda on lihtsam põhjendada kui tavapärast andmete sidumist. Miinuseks on jällegi see, et React hõlmab ainult vaate kihti ja arendamiseks peab valima teisi tehnoloogiaid lisaks. [19]

1.3.2 Vue

Vue või ka VueJs on progressiivne JavaScripti raamistik kasutajaliidese loomiseks. Erinevalt ühtsetest raamistikest on Vue kavandatud järk-järgult kasvavaks, rõhku on pandud vastuvõtlikkusele, mis tähendab et teiste moodulite ja raamistikega oleks seda lihtne siduda või integreerida. Jällegi sarnaselt React-ile on ka Vue keskendunud ainult vaatekihile ja kasutab virtuaalset DOM-i. [20]

1.3.3 Angular

Angular on TypeScripti komponedipõhine JavaScripti raamistik veebirakenduste loomiseks, mis on välja arendatud Google poolt. Arendusplatvorm sisaldab teekide kogumikku, mis omakorda hõlmab funktsioone nagu näiteks marsruutimine, kahepoolen andmete sidumine (ingl k *two-way data binding*), vormide haldamine, kliendi ja serveri vahelist suhtlust. Lisaks aitab raamistik lihtsustada programmeerimisel arendamist, koostamist, testimist ja uuendamist. [21]

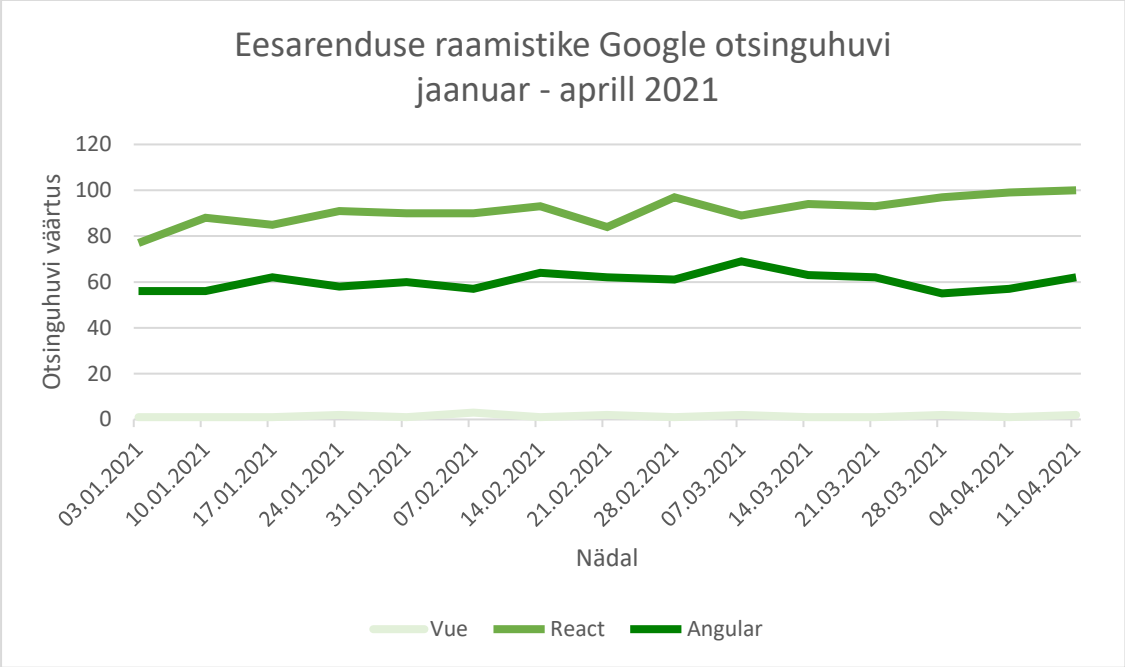
Angulari plussideks on andmete kahesuunaline sidumine, MVC komponendipõhine arhitektuur, populaarsus ja hästi kasutatavus. Miinusteks aga mahukus, aeglasem jõudlus, pidev uuenemine ja raskus seda õppida.

1.3.4 Eesrakenduse valiku põhjendus

Eesrakenduse raamistike React-i, Vue ja Angular-i vahel osutus loodava veebirakenduse valikuks Angular. Kuigi võrreldes teiste nagu React ja Vue raamistikega on Angular tunduvalt keerukam, kasutab ta ühe asemel kahesuunalist andmevoogu, MVC raamastikus ei keskendu ainult vaatekihile, vaid see on ka komponendipõhine. Angulari projektid on struktureeritud mooduliteks, komponentideks ja teenusteks. Igal Angulari rakendusel on vähemalt üks põhikomponent ja üks põhimoodul. [22]

Kõik eelnavalt nimetatud kolm raamistikku on populaarsed, väga aktiivselt välja töötatud, nad annavad regulaarselt välja uusi versioone ning hooldavad olemasolevaid.

Järgnevalt on näha statistika Google-i otsingukasutamise kohta alates 2021. aasta jaanuarist kuni aprillini (Joonis 1.1). Arvud näitavad otsinguhuvi suhet graafiku kõrgpunktiga kogumaailmas. Väärtus 100 on termini maksimaalne populaarsus. Väärtus 50 tähendab, et termini populaarsus on poole väiksem. Tulemus 0 tähendab, et termini kohta polnud piisavalt andmeid. Google Trendi kohaselt on React kõige populaarsem, talle järgneb Angular ja siis Vue. [23]



Joonis 1.1 Google Trends React-i, Vue ja Angular-i otsinguhuvi võrdlus [23]

Valiku põhjuseks võib välja tuua kogemuse. Angulari raamistikku on eelnevalt autori poolt kõige enam kasutatud. Angular on ka osa tuntud MEAN virnastst ehk kogumist, mis koosneb MongoDB-st, Express-ist, Angular-ist ja Node.js-s.

1.4 Andmebaas

Edasistes alapeatükkides võetakse vaatluse alla kolm andmebaasi, millega veebirakenduse andmeid säilitada, tõlgendada ja töödelda. PostgreSQL ja MySQL põhinevad SQL (ingl k *Structured Query Language*) ehk struktuurpäringukeele relatsioonmudelil andmebaasihaldus süsteemil aga MongoDB on mitte SQL-il põhinev andmebaasihaldus süsteemil. MongoDB on dokumentide andmebaas, mis salvestab andmeid JSON-laadsetesse dokumentidesse.

1.4.1 PostgreSQL

PostrgreSQL on avatud lähtekoodiga realitsioonide andmebaasisüteem, mis kasutab ja laiendab SQL-i koos funktsioonidega. PostgreSQL funktsionaalsus jagatakse andmetüüpide, andmete terviklikkuse, samaaegsuse ja jõudluse, töökindluse,

salvestuse ja taaste, turvalisuse, laiendatavuse, indeksite, tähemärkide toe ja otsingu osadesse. [24]

1.4.2 MySQL

MySQL on avatud relatsioonide andmebaaside haldussüsteem (ingl k *relational database management system* (RDBMS)). MySQL on kiire, töökindel, skaleeritav, hõlpsasti kasutatav ja platvormidevaheline. Veebirakenduses on kasutatakse RDBMS-i anmebaasiprogrammi (nagu MySQL), serveripoolset skriptimiskeelt (näiteks PHP-ga töötab MySQL väga hästi), SQL-i ja lehe kujundamist HTML-i ja CSS-i, et andmebaasist anmdeid näidata. [25]

MySQL on väga populaarne ja seda kasutavad Facebook, Twitter, Airbnb, Booking.com, Uber, GitHub, YouTube veebileheküljed, WordPress, Drupal, Joomla!, Contao sisuhaldussüsteemid. [25]

1.4.3 MongoDB

MongoDB on vajaliku päringu, indekseerimisega soovitud mastaapsuse, andmeid koodina hoidev, induitiivne dokumentide, objektorienteeritud andmebaas. MongoDB on NoSQL ehk mitte SQL-i keelel põhinev andmebaas. MongoDB salvestab andmeid paindlikesse, JSON-laadsetesse dokumentidesse, see tähendab, et väljad võivad dokumenditi erineda ja andmestruktuuri saab aja jooksul muuta. MongoDB-sse on sisseehitatud sellised funktsioonid nagu automaatne tõrkeotsing, horisontaalne skaleerimine ja võimalus asukohale andmeid määrata. Dokumendimudel kaardistab rakenduskoodis olevad objektid. MongoDB on lihtne, kiire, paindlik, mitmekülgne ja andmed on skeemivabad. [26]

1.4.3.1 Mongoose

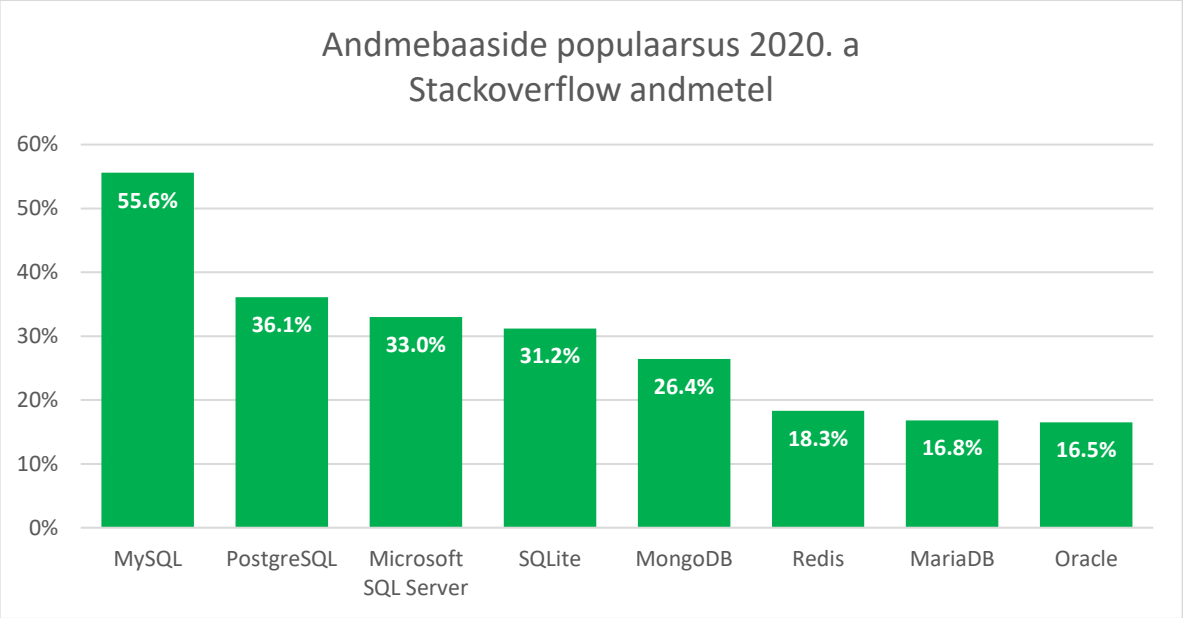
Mongoose on MongoDB objektide modelleerimise tööriist ehk ODM (ingl k *Object Document Mapper*), mis on kirjutatud JavaScripti abil ja mõeldud asünkroonses keskkonnas töötamiseks. [27]

Mongoose loodi MongoDB valideerimise, ülekande, äriloogika sidumise parendamiseks. Mongoose pakub rakenduse andmete modelleerimiseks sirgjoonelist skeemipõhist lahendust. See sisaldab sisseehitatud tüüpi ülekandmist, valideerimist, päringute koostamist ning äriloogika sidumist. [28]

1.4.4 Andmebaasi valiku põhjendus

Andmebaasisüteemide PostgreSQL-i, MySQL-i ja MongoDB-i vahel osutus loodava veebirakenduse valikuks MongoDB. Uuendusliku, mitmekülgsete funktsioonide ja poolstruktuuri poolest sobib MongoDB loodava veebirakenduse jaoks väga hästi. Lisaks on TalTech Virumaa kolledži andmebaasiserveris MongoDB olemas.

Järgnevalt on näha Stackoverflow 2020. aasta statistilised andmed (Joonis 1.2), mille kohaselt on kõigi 49537 vastunu arvates kõige populaarsem MySQL, teisel kohal PostgreSQL ja viiendal asub MongoDB [12].



Joonis 1.2 Stackoverflow 2020. aasta statistika andmebaaside populaarsusest

1.5 Tehnoloogia valiku põhjendus

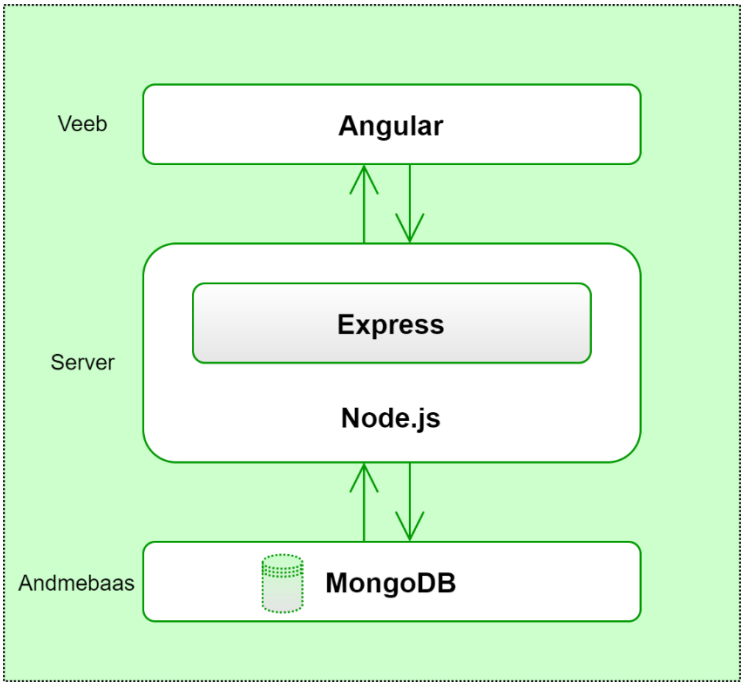
Node.js, Express, JWT autentimine, MongoDB ja Mongoose olid TalTech Virumaa kolledži poolse tellimussoovina kasutatavateks tehnoloogiateks. Asutuse andmebaasiserveris on PostgreSQL ja MongoDB andmebaasi võimalused ning sellest lähtuvalt sooviti MEAN virna (ingl k *stack*) kasutust. MEAN (*MongoDB, ExpressJS, AngularJS, and Node.js*) virn toimib ühtse tervikuna, välja arvatud JWT autentimine.

MEAN virn on ühtne JavaScripti põhine kihtkogum või teisisõnu pinu veebirakenduse arendamiseks. [29]

MEAN tähed jaotuvad järgnevalt [29]:

- MongoDB - anmdebaas
- Express(.js) - Node.js veebiraamistik
- Angular(.js) - JavaScripti kliendiosa raamistik
- Node(.js) - peamine JavaScripti veebiserver

Järgenavalt jooniselt on näha MEAN virna arhitektuur (Joonis 1.3) [29].



Joonis 1.3 MEAN virna arhitektuur

MEAN virna kuuluvad tehnoloogiad on populaarsed, avatud, lihtsasti kättesaadavad ja tasuta. Lisaks tuli tehnoloogia valikul tugineda ka mingilmääral kogemusele. Eelnevalt on autor eelnimetatud tehnoloogiaga töötanud ja see on juba tuttav. Teised võrdluses olnud realiseerimis meetoditega ei olnud autoril praktilise projekti loomise läbi eelnevat kogemust ja huvi.

2. VEEBIRAKENDUSE LOOMINE

Antud peatükis kirjeldatakse loodavat veebirakendust, selle sarnaseid lahendusi, nõudeid, loomis- või valmimisprotsessi ja struktuuri.

2.1 Veebirakenduse kirjeldus

Käesoleva rakenduskõrgharidusõppe lõputöö raames valmiv veebirakendus kannab nime VikoPub ja selle eesmärgiks on koondada ühte veebisüsteemi erinevad TalTech Virumaa kolledžiga seonduvad publikatsioonid. Projekt peab olema kättesaadav kinnise veebilehena, millele ligipääs on vaid kolledži töötajatel, kellele luuakse kasutajad. Lõppkasutajale peab veebileht olema lihtne ja arusaadav. Publikatsioonid peavad olema hästi kategoriseeritud ja organiseeritud. Kasutajate ja kategooriate lisamine ning kustutamine ja lisaks põhitegevused, milleks on artiklite lisamine, muutmine, vaatamine, kustutamine, peab olema kasutajale kiire ning mugav.

2.2 Analüüs ja arendusnõuded

Algselt uuriti olemasolevad publikatsioonide hallatavaid lahendusi või võrreldavaid veebisüsteeme. Mis on nende lahenduste puudusteks ja piiranguteks ning mida peab arvestama loodava VikoPub veebirakenduse puhul.

Tallinna Tehnikaülikool (TalTech) kasutab meediamonitooringu ja analüüsi teenust STATION (station.ee), mis hõlmab kogu TalTech-i asutuse infovooge.

Lisaks on olemas Eesti Teadusinfosüsteem - etis.ee (ETIS), kus on menüüs oleva Teadustegevuste all eraldi publikatsioonid. Asutuse järgi saab otsida - Tallinna Tehnikaülikool, Inseneriteaduskond, Virumaa kolledž. ETIS hõlmab endas eelkõige teaduslikke publikatsioone.

TalTech digikogus (digikogu.taltech.ee) on samuti palju lõputöid, artikleid ja väljaandeid, millest leiab kogu TalTech-iga seonduvaid publikatsioone.

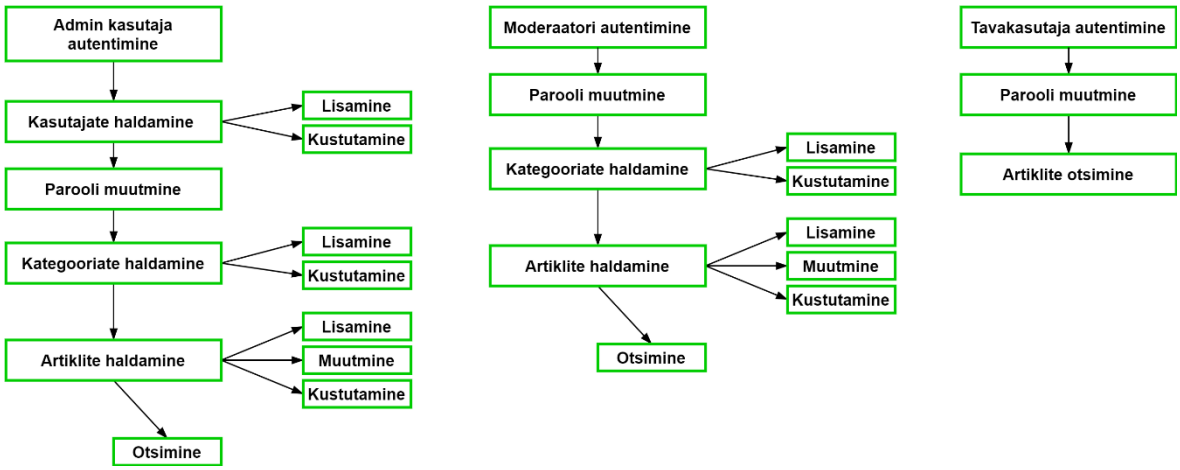
TalTech Virumaa kolledž kasutab eelnevalt mainitud veebisüsteeme aga rohke ja liigse info tõttu oleks parem kui kolledžil endal oleks ainult Virumaa kolledži publikatsioonide veebirakendus. Virumaa kolledžile loodav veebirakendus peaks olema ainult asutusesisene ja sisaldama lisaks teaduslikele artiklitele veel ka näiteks asutuse reklaami, sündmuste, isikute jms ehk kõiki valdkondi hõlmavaid, artikleid.

Täpsema ettekujutuse saamiseks, missugust veebirakendust, missuguse funktsionaalsusega, toimingute ja päringutega veebisüsteemi TalTech Virumaa kolledž soovib, viidi läbi intervjuu kolledži turundus- ja kommunikatsioonijuhi Annely Oonega, kes eeldatavasti hakkab veebisüsteemi kasutama. Kohtumise käigus selgitati välja loodava veebirakendusele esitatud nõuded, mis on konkreetsemalt välja toodud edasises loetelus.

Veebirakenduse nõuded

- Veebirakenduse kasutamiseks on vaja kasutajakontot
- Admin kasutaja peab saama vahetada oma parooli
- Admin kasutaja peab saama luua ja kustutada kõiki kasutajaid
- Kasutajate loomisel ja muutmisel peab olema võimalik redigeerida järgnevat infot:
    - Kasutajanimi
    - Parool
    - E-post
    - Rollid
- Admin kasutaja peab saama näha, otsida, lisada, muuta, kustutada publikatsioone.
- Admin kasutaja peab saama näha, lisada ja kustutada kategooriaid
- Moderaator peab saama näha, otsida, lisada, muuta, kustutada publikatsioone.
- Tavakasutaja peab saama näha ja otsida publikatsioone.
- Publikatsioonide loomisel ja muutmisel peab olema võimalik redigeerida järgnevat infot:
    - Pealkiri
    - Kuupäev
    - Autor
    - Väljaande tüüp (raadio, televisioon)
    - Seotud inimesed - nimed (intervjueeritav)
    - Seotud otsingu sõnad
    - Fail (pdf, jpg)
- Kategooriate loomisel ja muutmisel peab olema võimalik redigeerida järgnevat infot:
    - Kategooria nimetus
    - Kategooria kirjeldus
- Publikatsioone peab saama otsida pealkirja, kuupäeva, autori, väljaande tüübi, seotud inimeste ja sõnade järgi.
- Publikatsioonid peavad olema loetelus kategooriate all.

Veebirakenduse nõuetest tulenevalt on tehtud tegevustest ülevaade (Joonis 2.1).



Joonis 2.1 VikoPub veebirakenduse funktsionaalsus

2.3 Loomisprotsess

Enne kui veebirakenduse tegema hakkati, õpiti bezkoder.com veebilehel olevate õpetuste järgi, kuidas oleks võimalik veebilahendust realiseerida. Lõputöö veebirakendus põhineb eelnevalt nimetud keskkonna juhenditele ja selle loomiseks kasutatakse Java-s kirjutatud integreeritud tarkvararakendus keskkonda IntelliJ IDEA.

Tehnoloogiliselt luuakse MEAN virna osaline CRUD rakendus, milles tagarakenduse server kasutab Nodejs-i ja Express-i REST API jaoks, eesrakendus on Angulari rakendus HTTP-päringutega. Lisaks tehakse tagarakendusse Node.js Express koos JWT autentimine registreerimiseks ja sisselogimiseks ning nende vaated eesrakenduse projekti. CRUD (ingl k *create, read, update, delete* lühend) on antud veebirakenduses artiklite loomiseks, lugemiseks, muutmiseks ja kustutamiseks andmebaasiga suhtlev rakenduse tüüp. Lisaks artiklitele on vaja luua ka kategooriate ja kasutajate loomise ja kustutamise funktsioonid.

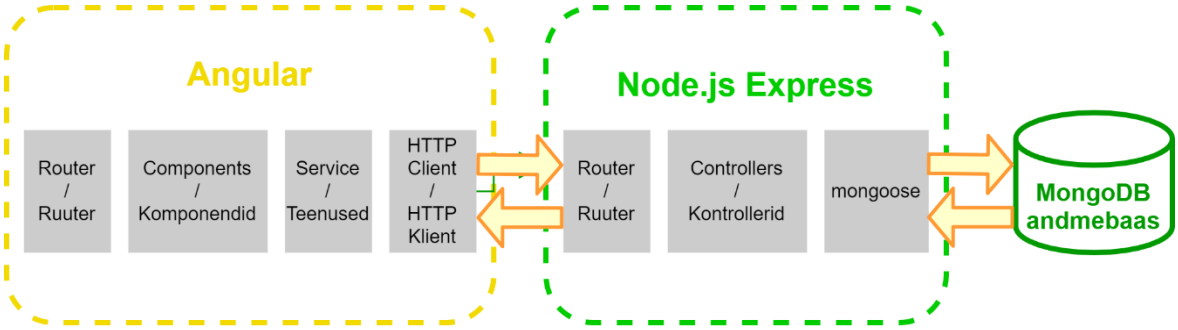
Tagarakendus ja eesrakendus hakkavad oma vahel funktsioneerima, andmeid vahetama ning on pidevas omavahelises suhtluses ja koostöös. Mõlemad tehtavad projektid käivituvad ühes ja samas kohas, mis tähendab, et Angulari ja Node.js teenused tuleb integreerida.

Kogu veebirakenduse avaliku veebilehe komponendid ja vaated hakkavad olema eesrakendus projektis, mille kaudu kasutaja erinevate funktsioonide käsklusi ja toiminguid edastab.

2.4 Veebirakenduse struktuur

Edasisestes alapeatükkides tutvustatakse veebirakenduse taga- ja eesrakendus struktuuri, koodinäiteid, marsruutimist, andmemudeleid, JWT autentimist, lühikest kirjeldust veebirakenduse käivitamise kohta ja viimaseks kujundusena vaateid.

Üldine MEAN virna CRUD rakenduse struktuur (Joonis 2.2) [30].

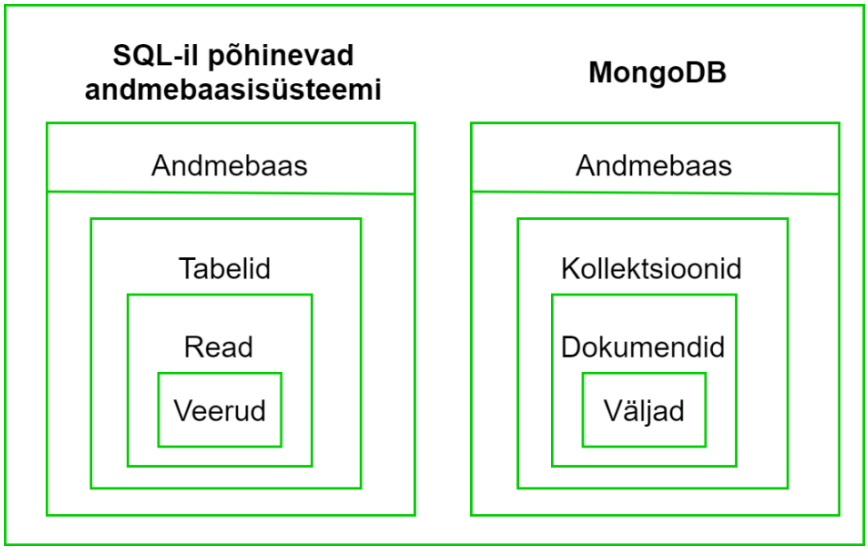


Joonis 2.2 MEAN virna CRUD rakenduse struktuur

Node.js Expressi osa koos MongoDB ühendusega jääb VikoPub tagarakenduse projekti nimetusega ArticleNodeExpress. Angular saadab HTTP-päringuid ja tagastab HTTP-vastuseid, kasutades HTTPClient-i [30]. VikoPub eesrakenduse projekt nimetusega ArticleAngular on Angulari osa.

MEAN virna veebirakenduse arenduseks on vaja installida ka vajalikud NPM (ingl k Node Package Manager) moodulid nagu body-parser, mis aitab JSON andmeid, teksti ja obketi liigendada, CORS (ingl k *cross-origin resource sharing*) on mehhanism, mille kaudu võimaldatakse veebilehele võtta puuduvaid ressursse teisest allikast [31], Express Node.js raamistik, mis aitab toetada veebirakenduse arendamist ja REST API-sid ehk arhitektuuri HTTPClient-i ja serverirakenduse vaheliseks suhtluseks ning mongoose ODM-i, millega suheldakse MongoDB andmebaasiga. Moodulite installimiseks kasutatakse käsklust *npm install* või täpsemalt just konkreetsete moodulite lisamiseks võib kasutada käsklusi *npm install express mongoose body-parser cors --save* ja *npm install -g @angular/cli*.

Kui tavaline relatsiooni ehk SQL-il põhinevas andmebaasis on tabelid, read, veerud siis MongoDB-s on kollektsioonid, mis koosnevad dokumendiväljadest. Järgnevalt on esitatud SQL-il põhinevate ja MongoDB andmebaaside struktuuriline erinevus (Joonis 2.3).

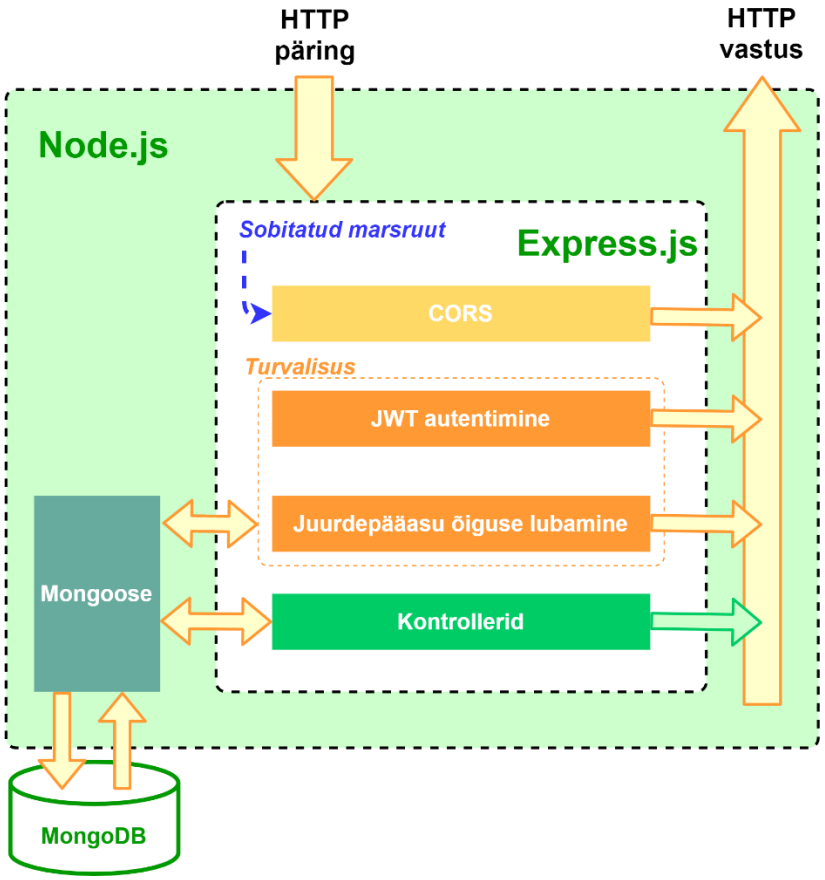


Joonis 2.3 SQL-il põhineva ja MongoDB andmebaasisüsteemide struktuuri võrdlus

VikoPub veebirakenduse andmed on salvestatud MongoDB andmebaasi. MongoDB kollektsioonid, dokumendid ja väljad on esitatud kasutajate, rollide, kategooriate ja artiklite kohta (Lisa 1).

2.4.1 Projekti tagarakenduse struktuur

Tagarakenduse Node.js Express võib kokkuvõtta järgneva struktuuriga (Joonis 2.4).



Joonis 2.4 Node.js Express tagarakenduse projekti struktuur

ArticleNodeExpress projektis on erinevad kaustad ja failid, mis on täpsemalt nimetatud allpool loeteludena. Alustuseks on projekti loodud *server.js* fail, milles on vajalike moodulite ja marsruutide importimine, lähtestamine ja ühenduste haldamine.

Kasutajaid, kategooriaid ja publikatsioone hoitakse MongoDB andmebaasis, kust suhtlemiseks kasutatakse mongoose-t.

ArticleNodeExpress projektis *app* kaust koosneb järgnevatest alamkaustades:

- *config* - sätete kaust, kus on failid:
    - *auth.config.js* - JWT salajase teksti võtme konfigureerimine, mis näeb välja järgnevalt: module.exports = { secret: "..."};
    - *db.config.js* - MongoDB andmebaasi ühendamise konfigureerimine, mis näeb välja nii: module.exports = { url: "..."};
- *controllers* - kontrollerite kaust, kus on failid:
    - *article.controller.js*
    - *auth.controller.js*
    - *category.controller.js*
    - *file.controller.js*
    - *user.controller.js*

Kontrollerid haldavad artiklitega, registreerimisega ja sisselogimisega, kategooriatega, failidega ja kasutajatega seonduvaid toiminguid. Näiteks artiklites on esindatud CRUD funktsioonid nagu loo (ingl k *create*), otsi kõik (ingl k *findAll*), otsi üks (ingl k *findOne*), uuenda (ingl k *update)* ja kustuta (ingl k *delete*).

Uue artikli loomise ja salvestamise toimingu koodinäide (Joonis 2.5).

```
const db = require("../models");
const Article = db.article;

// Create and Save a new Article
exports.create = (req, res) => {
    // Validate request
    if (!req.body.title) {
        res.status(400).send({ message: "Content can not be empty!" });
        return;
    }
```

```
    // Create a Article
    const article = new Article({
        title: req.body.title,
        dateCreated: req.body.dateCreated,
        description: req.body.description,
        author: req.body.author,
        category: req.body.category,
        articleType: req.body.articleType,
        relatedPeople: req.body.relatedPeople,
        searchWords: req.body.searchWords
    });

    // Save Article in the database
    article
        .save(article)
        .then(data => {
            res.send(data);
        })
        .catch(err => {
            res.status(500).send({
                message:
                    err.message || "Some error occurred while creating the Article."
            });
        });
};
```

Joonis 2.5 article.controller.js kood

- *models* - mudelite kaust, kus on failid:
    - *article.model.js*
    - *category.model.js*
    - *file.model.js*
    - *index.js*
    - *role.model.js*
    - *user.model.js*

Mongoose mudelite defineerimine (Joonis 2.6):

```
const dbConfig = require("../config/db.config.js");

const mongoose = require("mongoose");
mongoose.Promise = global.Promise;

const db = {};
db.mongoose = mongoose;
db.url = dbConfig.url;
```

```
db.article = require("./article.model.js");
db.category = require("./category.model.js");
db.user = require("./user.model");
db.role = require("./role.model");
db.ROLES = ["user", "admin", "moderator"];
db.file = require("./file.model.js");

module.exports = db;
```

Joonis 2.6 *index.js* (models)

connect() meetodi lisamine (Joonis 2.7):

```
const db = require("./app/models");
const Role = db.role;
const User = db.user;

db.mongoose
    .connect(db.url, {
        useNewUrlParser: true,
        useUnifiedTopology: true
    })
    .then(() => {
        console.log("Connected to the database!");
        initial();
    })
    .catch(err => {
        console.log("Cannot connect to the database!", err);
        process.exit();
    });
```

Joonis 2.7 *server.js*

Koodinäide, milles on mongoose mudeli näide (Joonis 2.8):

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema

let articleSchema = new Schema(
    {
        title: String,
        dateCreated: Date,
        description: String,
        author: String,
        files: [{
            type: Schema.Types.ObjectId,
            ref: "file"
        }],
        category: {
            type: Schema.Types.ObjectId,
            ref: "category"
        },
```

```
        articleType: String,
        relatedPeople: [ String ],
        searchWords: [ String ],
    },
    { timestamps: true }
);

articleSchema.method("toJSON", function() {
    const { __v, _id, ...object } = this.toObject();
    object.id = _id;
    return object;
});

module.exports = mongoose.model("article", articleSchema)
```

Joonis 2.8 articleSchema

Andmebaasi salvestatud artikli näide (Joonis 2.9). Mongoose mudel esitab MongoDB andmebaasi kollektsioonina artikli. Väljad luuakse automaatselt iga loodava artikli jaoks.


```
{
    "_id": {
        "$oid": "608ae3e14df6b11db8ab593b"
    },
    "files": [{
        "$oid": "609171e2790e381da0bca3cd"
    }],
    "relatedPeople": [],
    "searchWords": [],
    "title": "VIRUMAA KOLLEDŽI ÄRIINFOTEHNOLOOGIA MAGISTRIÕPPE INFOÕHTU",
    "dateCreated": {
        "$date": "2021-04-29T00:00:00.000Z"
    },
    "description": "TalTech Virumaa kolledži virtuaalne Äriinfotehnoloogia
    "author": "TalTech Virumaa kolledž",
    "category": {
        "$oid": "608ae2e64df6b11db8ab593a"
    },
    "articleType": "Internet",
    "createdAt": {
        "$date": "2021-04-29T16:50:41.140Z"
    },
    "updatedAt": {
        "$date": "2021-05-04T16:10:10.098Z"
    },
    "__v": 0
}
```


Joonis 2.9 MongoDB kollektsiooni näide

- *routes* - marsruutide kaust, kus on failid:
    - *article.routes.js*
    - *auth.routes.js*
    - *category.routes.js*
    - *upload.routes.js*
    - *user.routes.js*

Marsruutides on CRUD ja REST API-d, mille URL-id ja tegevused on esitatud osaliselt järgnevalt (Tabel 2.1).

Tabel 2.1 REST API veebiteenused

Meetod	**URL-id**	**Tegevused**
POST	api/auth/signin	sisselogimisel õigete kasutajaandmetega, tagastab rollid ja tagarakenduse poolt genereeritud JWT loa (ingl k *token*).
GET	api/auth/admin	kontrollib JWT loa põhjal, kas tegemist on admin rollis oleva kasutajaga
POST	api/category	lisab uue kategooria
GET	api/category	tagastab kõik kategooriad
DELETE	api/category/:id	kustutab ühe kategooria id järgi
POST	api/articles	lisab uue artikli
GET	api/articles	tagastab kõik artiklid
GET	api/articles/:id	tagastab ühe artikli id järgi
PUT	api/articles/:id	uuendab/muudab ühte artiklid id järgi
DELETE	api/articles/:id	kustutab ühe artikli id järgi
...		

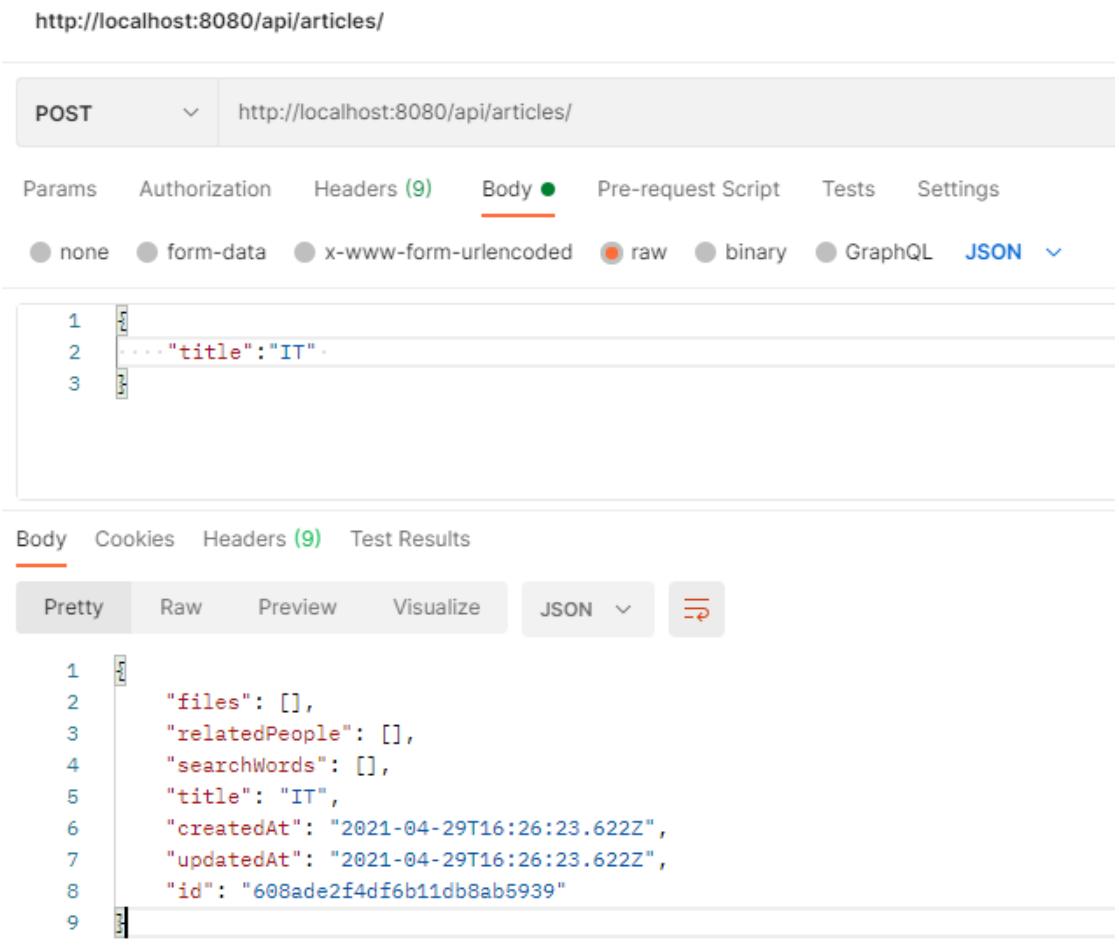
Marsruutimise POST, GET, PUT ja DELETE meetodite kasutamine artiklite puhul on nähtav järgneva koodinäitena (Joonis 2.10).

```
const { auth } = require("../services");

module.exports = app => {
    const article = require("../controllers/article.controller.js");

    let router = require("express").Router();

    // Create a new Article
    router.post("/", [auth.verifyToken, auth.isModeratorOrAdmin], article.create);

    // Retrieve all Articles
    router.get("/", [auth.verifyToken], article.findAll);

    // Retrieve a single Article with id
    router.get("/:id", [auth.verifyToken, auth.isModeratorOrAdmin], article.findOne);
```

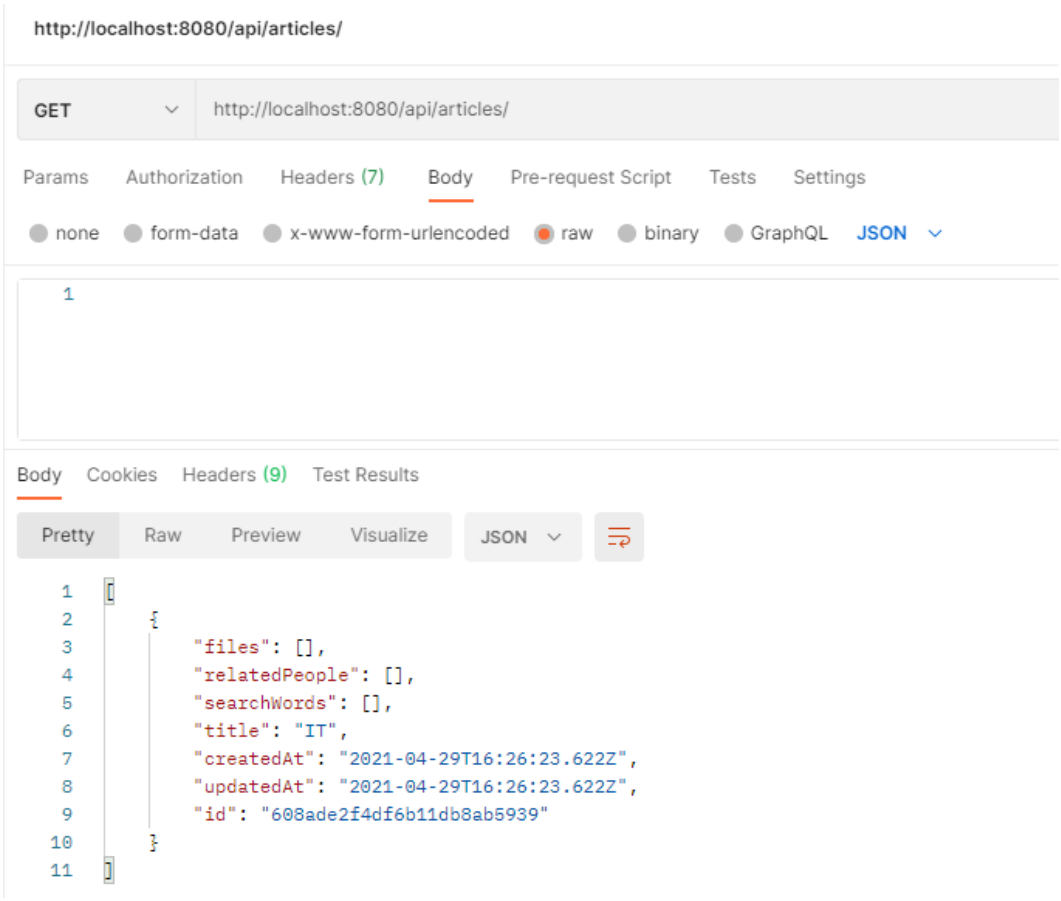
```
    // Update a Article with id
    router.put("/:id", [auth.verifyToken, auth.isModeratorOrAdmin], article.update);

    // Delete a Article with id
    router.delete("/:id", [auth.verifyToken, auth.isModeratorOrAdmin], article.delete);

    app.use("/api/articles", router);
};
```

Joonis 2.10 *article.routes.js*

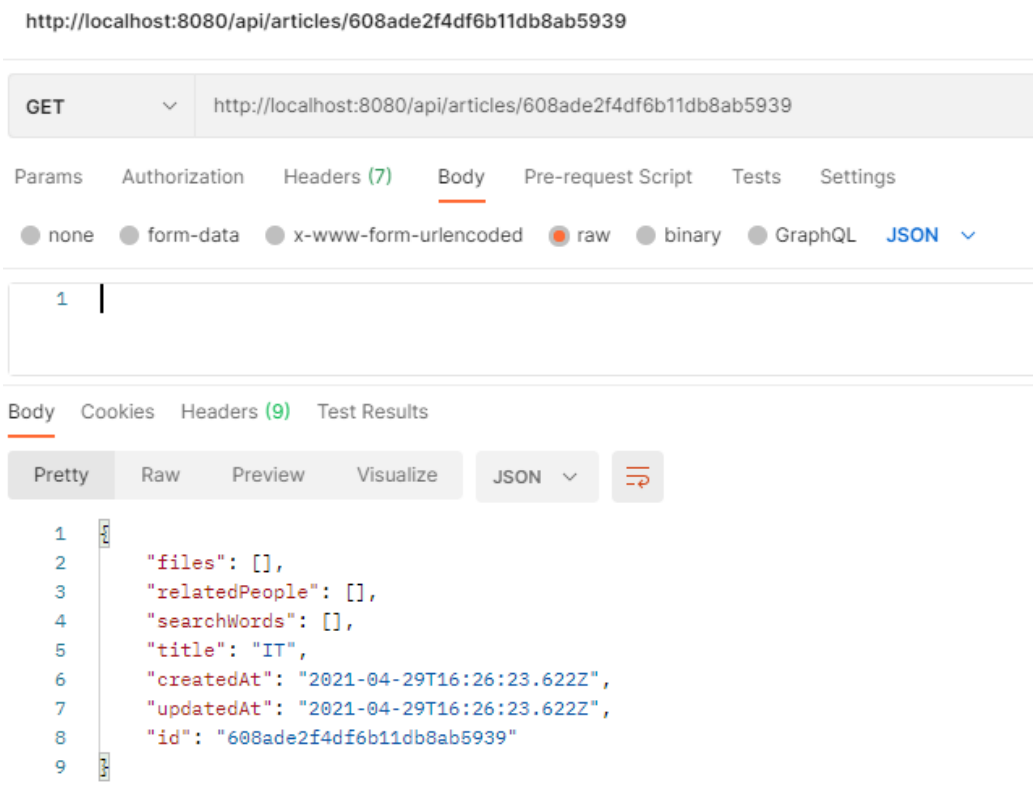
Marsuutimise tegevuste katsetamiseks viidi läbi testimine Postman-i abil. Postman on API arendamise platvorm, kus saab testida näiteks REST API päringuid. VikoPub tagarakenduse testimise API-d, mis on artiklite tegevused, on esitatud järgnevalt (Joonised 2.11 - 2.14).



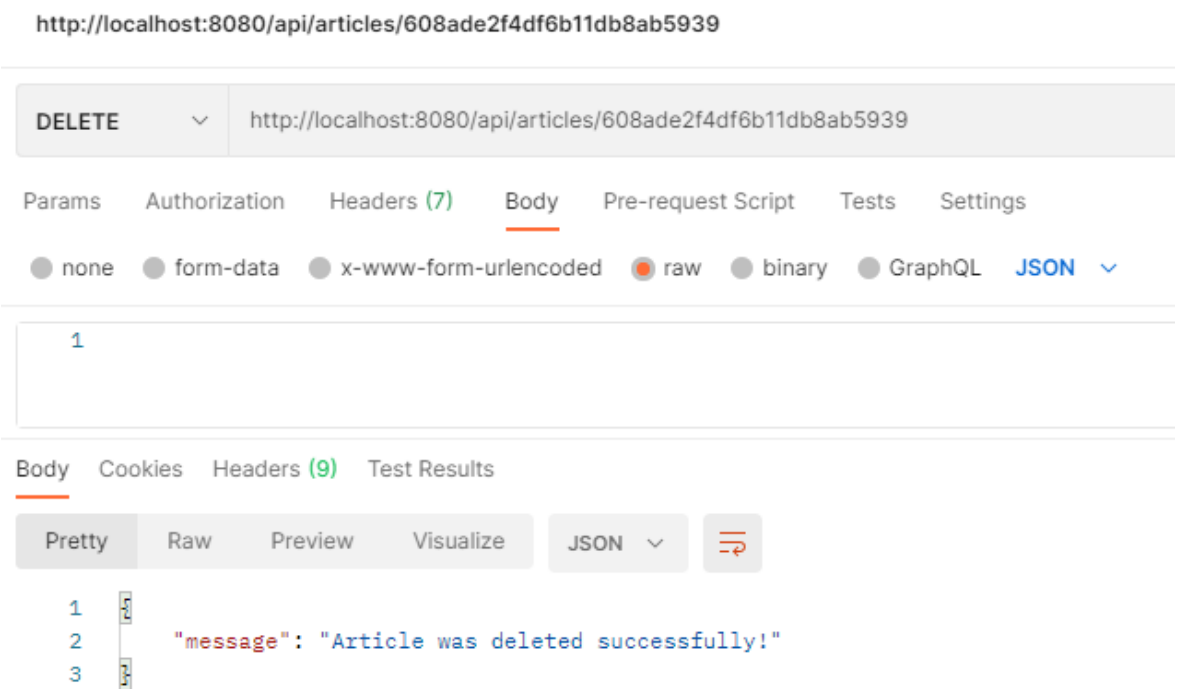
Joonis 2.11 POST päringu tegemisel lisati uus artikkel



Joonis 2.12 GET päringuga tagastati kõik artiklid (andmebaasis üks atrikkel)



Joonis 2.13 GET :id päringuga tagastati id järgi üks artikkel



Joonis 2.14 DELETE :id päringuga kustutab id järgi atrikli

- *services* - teenuste kaust, kus on failid:
    - *auth.service.js*
    - *index.js*
    - *upload.service.js*
    - *verify.service.js*

Teenused kontrollivad registreerumisel, kas olemasolev kasutajanimi või e-posti aadress on juba kasutusel, JWT luba ja andmebaasis kasutajarolle. Näitena on faili *verify.service.js* kood (Joonis 2.15).

```
const db = require("../models");
const ROLES = db.ROLES;
const User = db.user;

checkDuplicateUsernameOrEmail = (req, res, next) => {
    // Username
    User.findOne({
        username: req.body.username
    }).exec((err, user) => {
        if (err) {
            res.status(500).send({ message: err });
            return;
        }

        if (user) {
            res.status(400).send({ message: "Failed! Username is already in use!" });
            return;
        }
```

```
        // Email
        User.findOne({
            email: req.body.email
        }).exec((err, user) => {
            if (err) {
                res.status(500).send({ message: err });
                return;
            }

            if (user) {
                res.status(400).send({ message: "Failed! Email is already in use!" });
                return;
            }

            next();
        });
    });
};

checkRolesExisted = (req, res, next) => {
    if (req.body.roles) {
        for (let i = 0; i < req.body.roles.length; i++) {
            if (!ROLES.includes(req.body.roles[i])) {
                res.status(400).send({
                    message: `Failed! Role ${req.body.roles[i]} does not exist!`
                });
                return;
            }
        }
    }

    next();
};

const verifyService = {
    checkDuplicateUsernameOrEmail,
    checkRolesExisted
};

module.exports = verifyService;
```

Joonis 2.15 *verify.service.js*

ArticleNodeExpress projektis on ka *resources* kaust, kus on alamkaust *uploads* ja sinna salvestatakse artiklitele muutmise käigus lisatud failid.

2.4.2 JWT Authentication

JSON (ingl k *JavaScript Object Notation*) JavaScripti objektide notatsioon. XML alternatiiv, avatud inimloetava teksti standardvorming (RFC 7159, ECMA-404) atribuudi

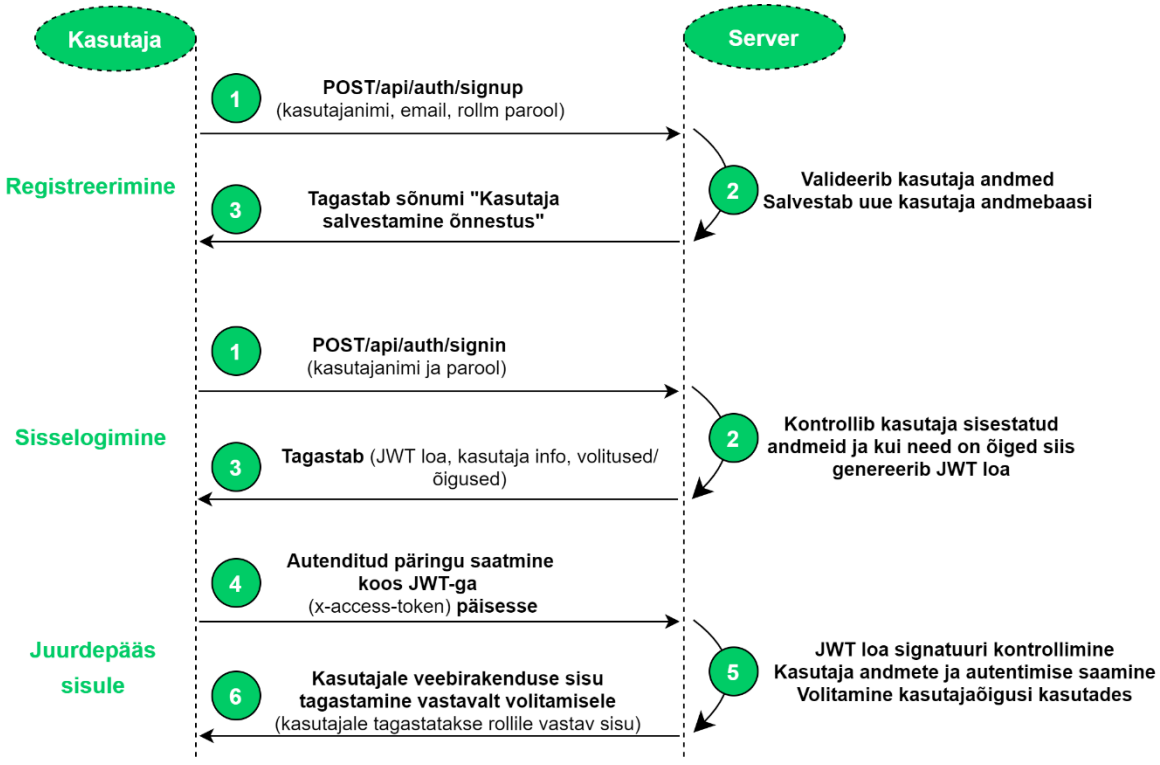
ja väärtuse paaridest koosnevate andmeobjektide edastuseks, eeskätt serveri ja veebirakenduse vahel; põhineb JavaScripti alamhulgal, kuid ei sõltu keelest. [32]

JWT (ingl k *Json Web Token*) on luba (RFC 7159), mis määrab komplektse ja iseseisva viisi info turvaliseks edastamiseks osapoolte vahel JSON-objektina.

JWT luba (ingl k *token*) koosneb kolmest osast:

1. Päis (ingl k *header*), milles on JSON kujul räsialgoritm, mida kasutatakse signatuuri valideerimiseks.
2. Andmed (ingl k *payload*) on näiteks kasutaja nimi, email.
3. Signatuur (ingl k *signature*) on päise ja andmete räsi. [33]

VikoPub veebirakenduses kasutatakse JWT luba kasutajate registreerimisel, sisselogimisel ja autoriseerimisel (Joonis 2.16). JWT on valitud oma tugeva turvalisuse ja kompaktsuse pärast. Iga 24 tunni tagant uuendatatkse JWT luba. Kasutuses on JSON dokumente, mida on just eesrakenduses lihtne töödelda.



Joonis 2.16 JWT loa kasutamine VikoPub veebirakenduses

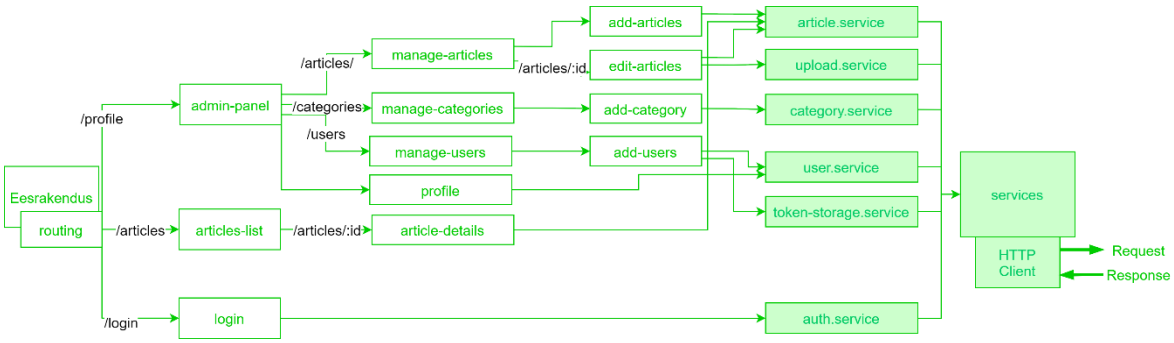
2.4.3 Projekti eesrakenduse struktuur

ArticleAngular projektis on erinevad kaustad ja failid, mis on täpsemalt nimetatud allpool loeteludena ja täpsustavate kirjeldustena.

Angulari rakendustes on peamisteks osadeks komponendid, millele on lisatud mallid. Komponendid tegelevad üldiselt rakenduse loogikaga ja vaadetega. Komponentide

kaustade ja failide loomiseks kasutatakse käsklust *ng g c component*/komponendi nimi, mis genereerib komponendi nimega kausta ja sinna sisse komponendi nimi*.component.css*, komponendi nimi*.component.html,* komponendi nimi*.component.ts* ja komponendi nimi*.component.spec.ts* failid. CSS (ingl k *cascading style sheets*) failis on defineeritud elementide kujundus. HTML failis on veebilehe osade, nagu näiteks tekstide, nuppude, lahtrite jms sisu määratlemine ning kus ja kuidas neid osi näidatakse. TypeScript-i failis on veebilehega seonduvate tegevuste ja funktsioonide esitamine. *.spec.ts failid on komponentide automaatseks testimiseks ja neid igas komponendi kaustas ei ole. Mudelid seovad funktsionaalsete üksuste moodustamiseks komponendid teenustega. VikoPub veebirakenduse jaoks on loodud eraldi komponentide kaust ja seal omakorda eelnevalt seletatud failid.

Eesrakenduse peamised tegevused on komponentide vahel funktsioneeritud marsruutidega (Joonis 2.17).



Joonis 2.17 Eesrakenduse marsruudid ja komponendid

ArticleAngular projektis on *src* kaust, kus on alamkaust *app* ning selles peamised failid:

- *app.component.css*
- *app.component.html*
- *app.component.spec.ts*
- *app.component.ts*
- *app.module.ts* - mis deklareerib Angulari koponendid ja impordib vajalikud moodulid.
- *app-routing.module.ts* - kus on defineeritud marsruudid, mis aitavad vaadete vahel navigeerimisi määratleda.

Lisaks on *app* kaustas omakorda alamkaustad:

- *components* - komponendid, kus on kasutad:
    - *admin-panel*
        - *components*

- *manage-articles …* (… tähendab, et sisaldab veel alamkaustu ja faile)
- *manage-categories …*
- *manage-users …*
- *profile …*

- *admin-panel.components.css*
- *admin-panel.components.html*
- *admin-panel.components.ts*
- *admin-panel.module.ts*

- *article-list*

Eesrakenduses on komponete väga palju aga koodinäited on järgnevalt välja toodud vaid sisselogimise kohta.

- *login*
  - *login.component.css* (Joonis 2.18) - sisselogimise paneeli kujundus.

```
label {
  display: block;
  margin-top: 10px;
}

.card-container.card {
  max-width: 400px !important;
  padding: 40px 40px;
}

.card {
  background-color: #f7f7f7;
  padding: 20px 25px 30px;
  margin: 50px auto 25px;
  -moz-border-radius: 2px;
  -webkit-border-radius: 2px;
  border-radius: 2px;
  -moz-box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
  -webkit-box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
  box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
}

.profile-img-card {
  width: 96px;
  height: 96px;
  margin: 0 auto 10px;
  display: block;
  -moz-border-radius: 50%;
  -webkit-border-radius: 50%;
  border-radius: 50%;
}
```

Joonis 2.18 *login.component.css*

- *login.component.html* (Joonis 2.19) - lahtrite, nupuude, tekstide asetsus ja määratlemine, kuidas neid on veebilehel olemas.

```
<div class="col-md-12">
  <div class="card card-container">
    <img
      id="profile-img"
      src="//ssl.gstatic.com/accounts/ui/avatar_2x.png"
      class="profile-img-card"
    />
    <form
      *ngIf="!isLoggedIn"
      name="form"
      (ngSubmit)="f.form.valid && onSubmit()"
      #f="ngForm"
      novalidate
    >
      <div class="form-group">
        <label for="username">Kasutajanimi</label>
        <input
          type="text"
          class="form-control"
          id="username"
          name="username"
          [(ngModel)]="form.username"
          required
          #username="ngModel"
        />
        <div
          class="alert alert-danger"
          role="alert"
          *ngIf="f.submitted && username.invalid"
        >
          Kasutajanimi on kohustuslik väli!
        </div>
      </div>
      <div class="form-group">
        <label for="password">Parool</label>
        <input
          id="password"
          type="password"
          class="form-control"
          name="password"
          [(ngModel)]="form.password"
          required
          minlength="5"
          #password="ngModel"
        />
        <div
          class="alert alert-danger"
          role="alert"
          *ngIf="f.submitted && password.invalid"
        >
```

```
            <div *ngIf="password.errors.required">Parool on kohustuslik väli</div>
            <div *ngIf="password.errors.minlength">
              Parooli pikkus peab olema 5 tähemärki
            </div>
          </div>
        </div>
        <div class="form-group">
          <button class="btn btn-outline-primary btn-block">
            Login
          </button>
        </div>
        <div class="form-group">
          <div
            class="alert alert-danger"
            role="alert"
            *ngIf="f.submitted && isLoginFailed"
          >
            Sisselogimine ebaõnnestus
          </div>
        </div>
      </form>

      <div class="alert alert-success" *ngIf="isLoggedIn">
        Olete sisselogitud.
      </div>
    </div>
  </div>
</div>
```

Joonis 2.19 *login.component.html*

- *login.component.ts* (Joonis 2.20) - funktsioonide ja tegevuste esitamine (JWT loa, kasutaja sisestuse kontroll)

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from 'src/app/services/auth.service';
import { TokenStorageService } from 'src/app/services/token-storage.service';
import { Router} from '@angular/router';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  form: any = {};
  isLoggedIn = false;
  isLoginFailed = false;
  roles: string[] = [];

  constructor(private authService: AuthService,
              private tokenStorage: TokenStorageService,
              private router: Router) { }
```

```
  ngOnInit(): void {
    if (this.tokenStorage.getToken()) {
      this.isLoggedIn = true;
      this.roles = this.tokenStorage.getUser().roles;
      setTimeout(() => {
        this.router.navigate(['/articles']);
      }, 1000);
    }
  }

  onSubmit(): void {
    this.authService.login(this.form).subscribe(
      data => {
        this.tokenStorage.saveToken(data.accessToken);
        this.tokenStorage.saveUser(data);

        this.isLoginFailed = false;
        this.isLoggedIn = true;
        this.roles = this.tokenStorage.getUser().roles;
        this.reloadPage();
      },
      err => {
        this.isLoginFailed = true;
      }
    );
  }


  reloadPage(): void {
    window.location.reload();
  }

}
```

Joonis 2.20 *login.component.ts*

- *models* - mudelis, kus on failis:
    - *article.ts* (Joonis 2.21) - artiklite parameetsite tüüpide määratlemine

```
import {ICategory} from './category';
import {IFile} from './file';

export interface IArticle {
  id?: string;
  title: string;
  category: ICategory;
  dateCreated: Date | string;
  dateUpdated?: Date | string;
  articleType: string;
  author: string;
```

```
  description: string;
  relatedPeople: string[];
  searchWords: string[];
  files: IFile[];
}
```

Joonis 2.21 *article.ts*

- *category.ts*
- *file.ts*
- *user.ts*

services - teenused kausta loomiseks kasutatakse käslust *ng g s services*/nimetus. Teenustes kasutatakse Angulari HTTPClient-it, et saata HTTP päringuid. Teenuste kasutas on failid:

- *article.service.ts* (Joonis 2.22) - funktsioonides kasutatakse CRUD toiminguid ja meetodeid.

```
import { Injectable } from '@angular/core';
import {HttpClient, HttpHeaders} from '@angular/common/http';
import { Observable } from 'rxjs';
import {TokenStorageService} from './token-storage.service';
import {environment} from '../../environments/environment';

const baseUrl = environment.apiUrl + '/api/articles';

@Injectable({
  providedIn: 'root'
})
export class ArticleService {

  constructor(private http: HttpClient, private tokenStorageService: TokenStorageService) { }

  authHeader = new HttpHeaders({ 'Content-Type': 'application/json', 'x-access-token'
  : this.tokenStorageService.getToken()});

  getAll(): Observable<any> {
    return this.http.get(baseUrl, { headers: this.authHeader });
  }

 get(id): Observable<any> {
   return this.http.get(`${baseUrl}/${id}`, { headers: this.authHeader });
 }

 create(data): Observable<any> {
   return this.http.post(baseUrl, data, { headers: this.authHeader });
 }

 update(id, data): Observable<any> {
   return this.http.put(`${baseUrl}/${id}`, data, { headers: this.authHeader });
 }
```

```
  delete(id): Observable<any> {
    return this.http.delete(`${baseUrl}/${id}`, { headers: this.authHeader });
  }

  findBySearchCriteria(searchField, searchWord): Observable<any> {
    return this.http.get(`${baseUrl}?searchCondition=${searchField}&searchString=${searchWord}`,
    { headers: this.authHeader });
  }
}
```

Joonis 2.22 *article.service.ts*

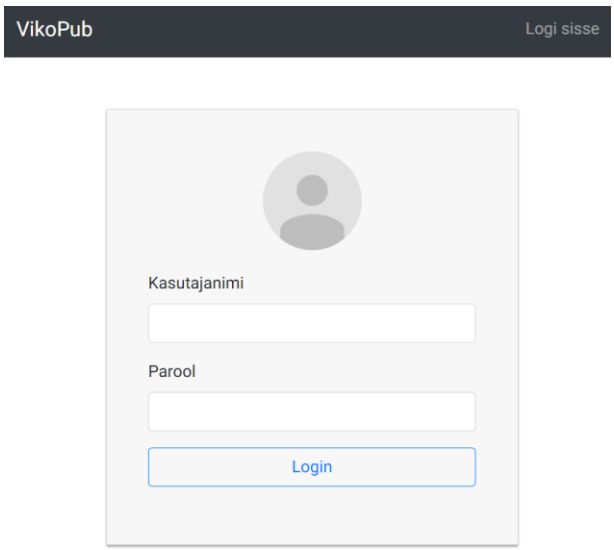
auth.guard.ts

- *auth.service.ts*
- *category.service.ts*
- *token/storage.service.ts*
- *upload.service.ts*
- *user.service.ts*

2.4.4 Veebirakenduse vaated

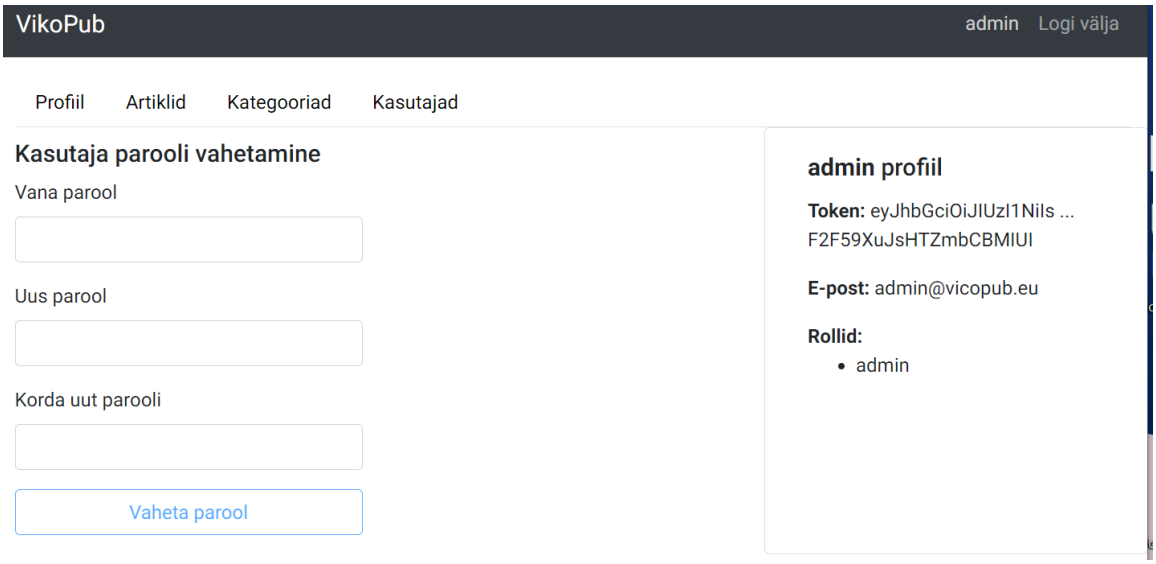
Järgnevalt on esitatud veebirakenduse vaated, mida admin kasutaja näeb ja saab redigeerida. Admin kasutaja vaated on esitatud sellepärast, et antud rollis on kõige rohkem õigusi ja see läbi saab kõige parema ülevaate VikoPub veebirakenduses olevatest haldamise võimalustest.

Joonisel 2.23 on sisselogimise vaade, mis on esimeseks vaateks kui veebirakenduse URL avatakse.



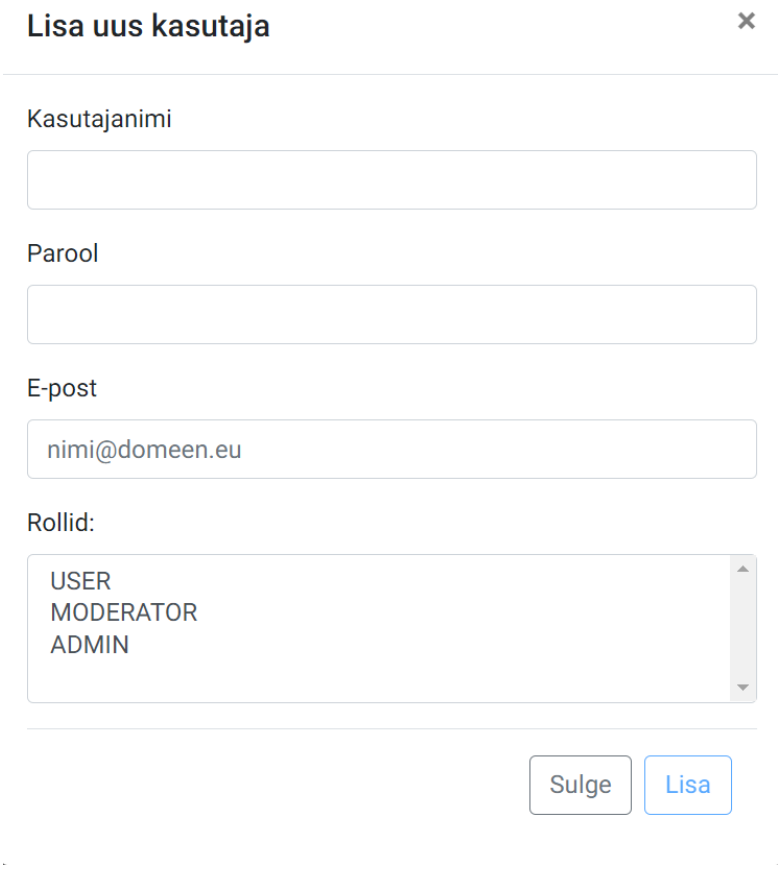
Joonis 2.23 Sisselogimine

Joonisel 2.24 on sisselogitud kasutaja ehk admin kasutaja profiilivaade kui vajutada admin ehk kasutaja nimetusel. Profiilil on võimalik enda kasutaja parooli vahetada.



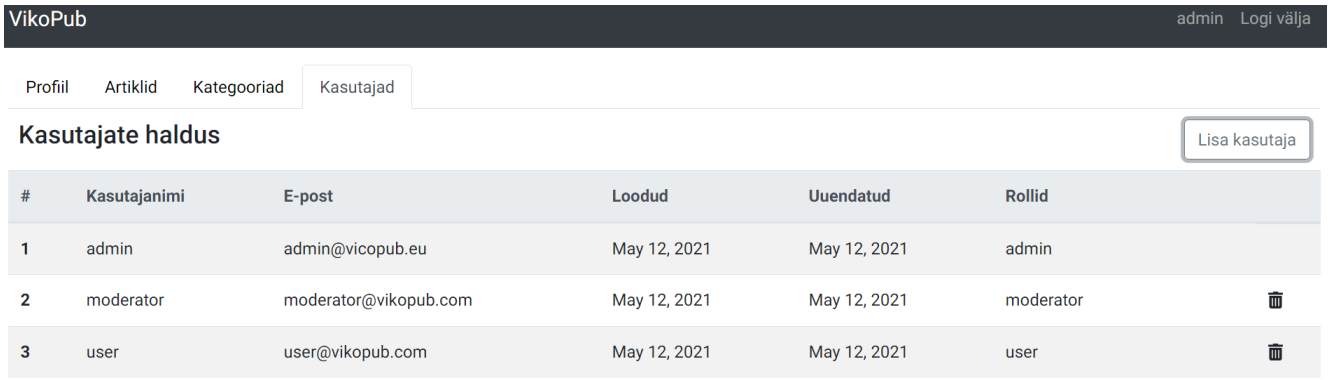
Joonis 2.24 Parooli muutmine

Vajutades Kasutajad ning Lisa kasutaja näeb uue kasutaja lisamise vaadet. Joonisel 2.25 on kasutaja lisamine või registreerimine, mis on veebirakenduses ainult admin kasutaja funktsiooniks.



Joonis 2.25 Kasutaja lisamine

Joonisel 2.26 on loodud kasutajate loetelu, mida saab avada Kasutajad alt.

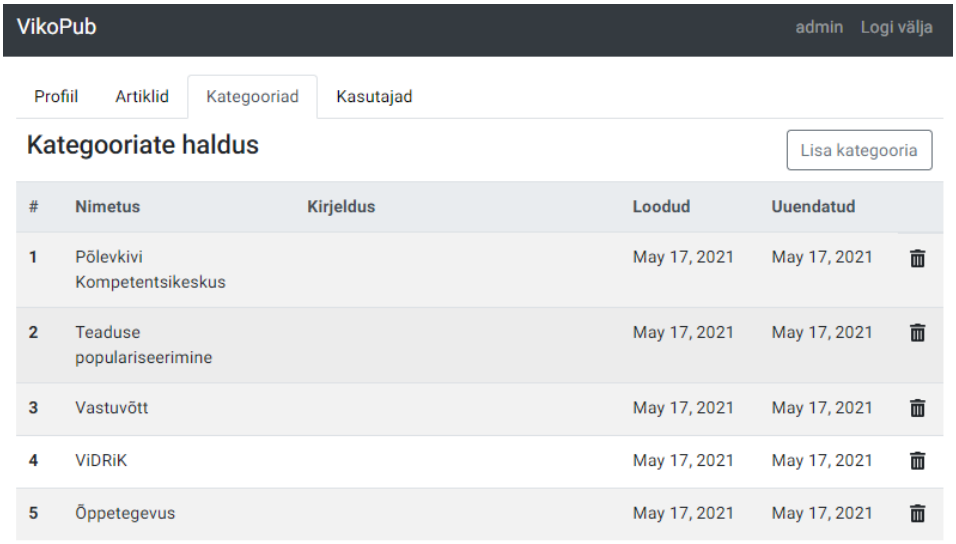


Joonis 2.26 Kasutajate loetelu

Joonisel 2.27 on kategooria lisamise vaade (Kategooriad -> Lisa kategooria).

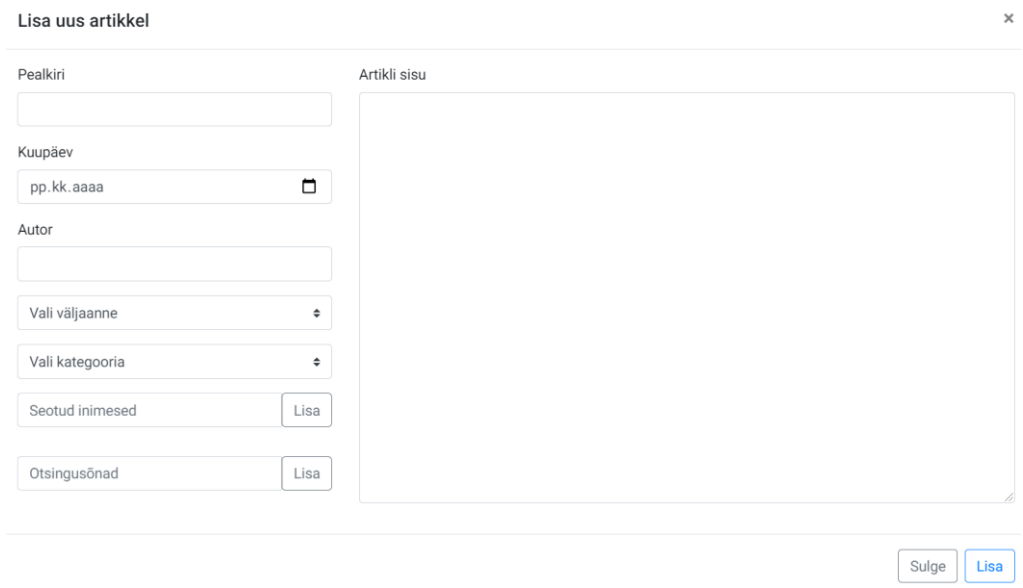
Joonis 2.27 Kategooria lisamine

Joonisel 2.28 on kateroogiate loetelu, mida saab avada Kategooriad alt.



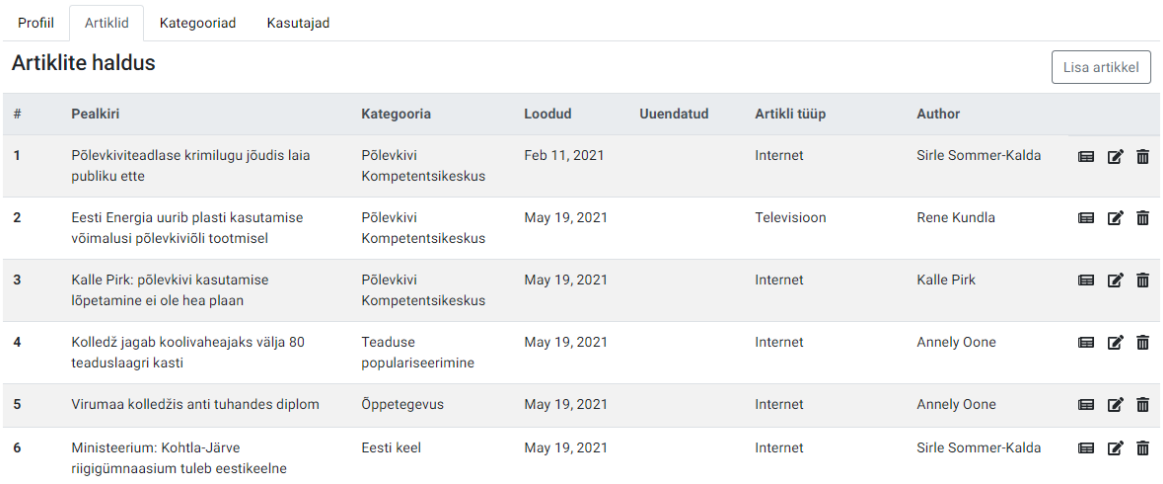
Joonis 2.28 Kategooriate loetelu

Joonisel 2.29 on artikli lisamise vaade (Artiklid -> Lisa artikkel).



Joonis 2.29 Artikli lisamine

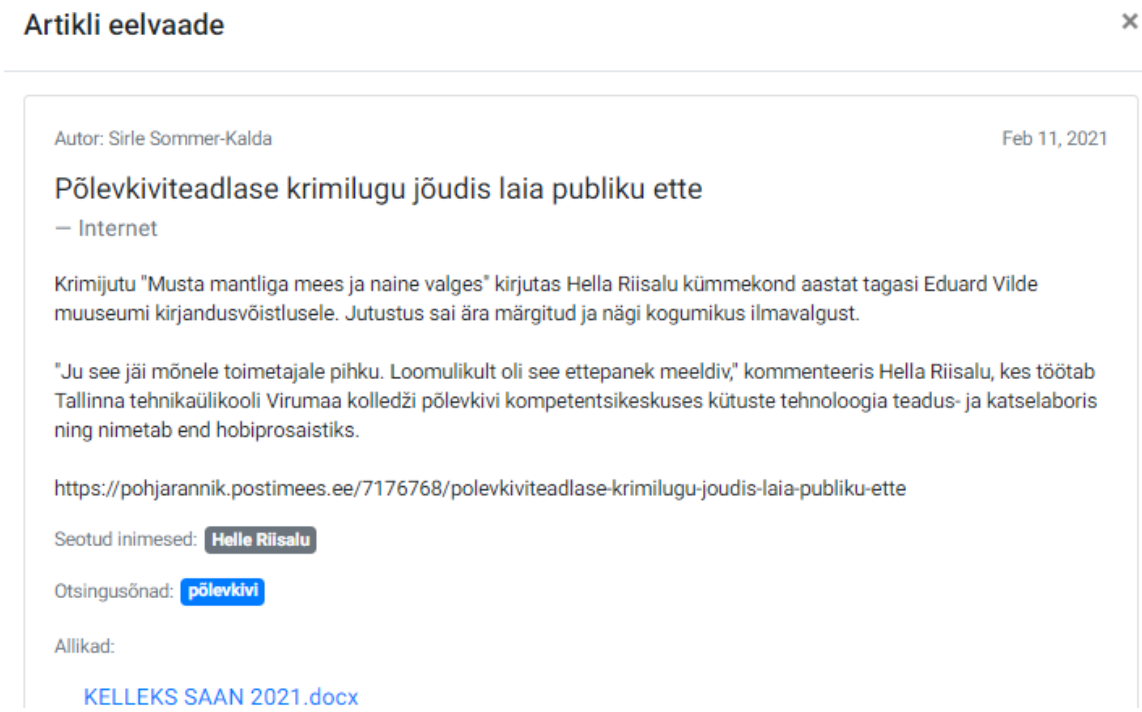
Joonisel 2.30 on artiklite loetelu, mida saab avada Artiklid alt.



#	Pealkiri	Kategooria	Loodud	Uuendatud	Artikli tüüp	Author
1	Põlevkiviteadlase krimilugu jõudis laia publiku ette	Põlevkivi Kompetentsikeskus	Feb 11, 2021		Internet	Sirle Sommer-Kalda
2	Eesti Energia uurib plasti kasutamise võimalusi põlevkiviõli tootmisel	Põlevkivi Kompetentsikeskus	May 19, 2021		Televisioon	Rene Kundla
3	Kalle Pirk: põlevkivi kasutamise lõpetamine ei ole hea plaan	Põlevkivi Kompetentsikeskus	May 19, 2021		Internet	Kalle Pirk
4	Kolledž jagab koolivaheajaks välja 80 teaduslaagri kasti	Teaduse populariseerimine	May 19, 2021		Internet	Annely Oone
5	Virumaa kolledžis anti tuhandes diplom	Õppetegevus	May 19, 2021		Internet	Annely Oone
6	Ministeerium: Kohtla-Järve riigigümnaasium tuleb eestikeelne	Eesti keel	May 19, 2021		Internet	Sirle Sommer-Kalda

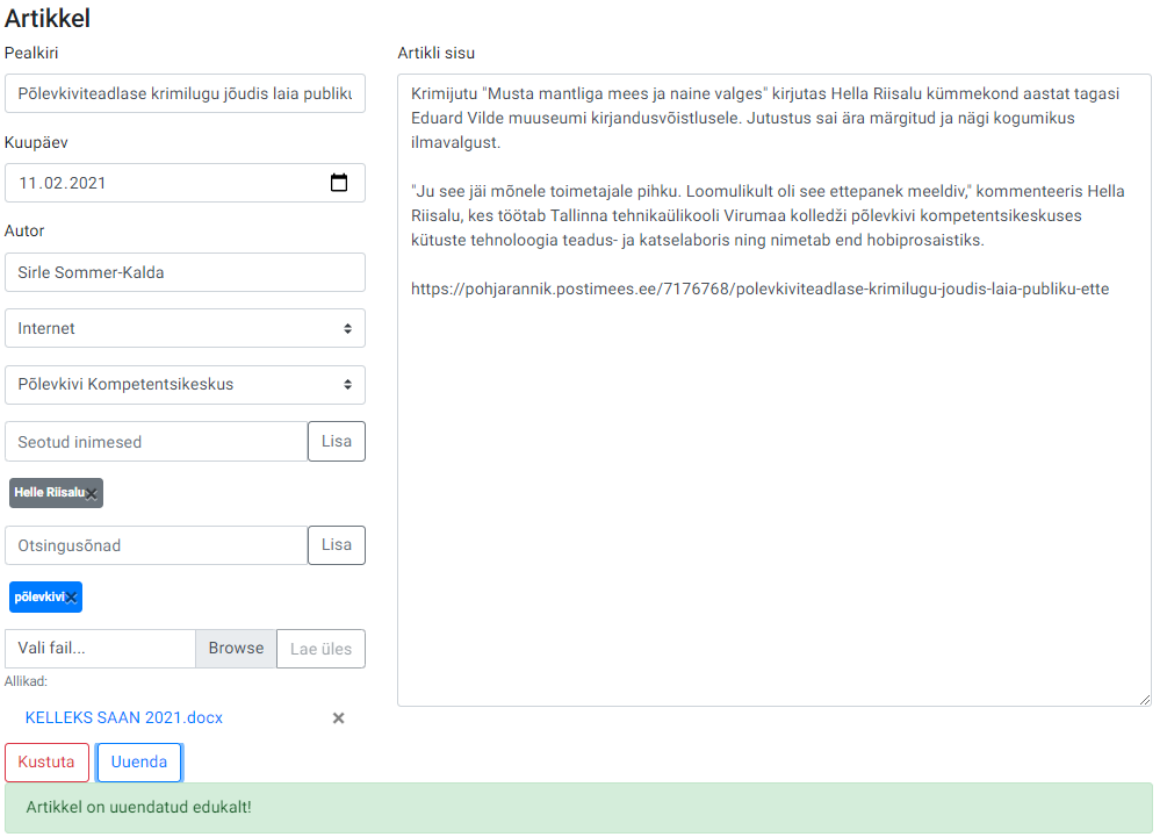
Joonis 2.30 Artiklite loetelu

Joonisel 2.31 on ühe valitud artikli eelvaade.



Joonis 2.31 Artikli eelvaade

Joonisel 2.32 on ühe valitud artikli muutmise vaade, kus saab parameetreid muuta ja faile lisada.



Joonis 2.32 Artikli muutmine

3. TULEMUS JA EDASIARENDAMINE

Selles peatükis kirjeldatakse veebirakenduse tulemust, esimest katsetamist eeldatava edaspidise kasutajaga ja lisaks tuuakse välja võimalused edasiarenduseks.

Lõputöö ülesanneteks oli uurida sarnaseid lahendusi, programme ja tehnoloogiaid ning neid omavahel võrrelda; teha REST API veebiteenused; kasutada MongoDB andmebaasi; luua JWT autentimine ja luua eesrakendus ja UI - kasutajaliides. Esimese peatükis on võrreldud erinevaid tehnoloogiaid ja kirjeldatud MEAN virna valikut tellimussoovina TalTech Virumaa kolldži poolt. REST API veebiteenused, MongoDB andmebaasi kasutamine, JWT autentimine ja eesrakenduse ning UI-kausutajaliidese on esitatud teises peatükis veebirakenduse loomine osas.

Valminud veebirakenduse kättesaadavaks tegemiseks on kasutatud Heroku pilveteenuse platvormi ja VikoPuv asub aadressil viko-pub.herokuapp.com. Veebileht sisaldab artikkleid, mis ei avalikult kättesaadavad ja sellest tingivalt on VikoPub kinnine ehk veebilehel nõutakse koheselt sisselogimist.

VikoPub-is on kolm kasutaja rolli, kelleks on admin kasutaja, moderaator ja tavakasutaja. Adminil on kõige rohkem võimalusi ja funktsioone, milleks on kasutajate, kategooriate, artiklite haldamine. Moderaatoril on kategoorite ja artiklite haldus. Tavakasutaja näeb ja saab otsida artikleid aga neid muuta ei saa.

Hinnang valminud tööle on hea. VikoPub veebirakenduse arendus toimus järk-järgult ja sujuvalt. Loomist alustati jaanuaris 2021. aastal ja peamiste funktsioonidega valmis rakendus mais 2021. aastal. Tellimusesoovina, kasutada MEAN virna ja JWT tehnoloogiat, sobis VikoPub-i teostamiseks väga hästi. Lõputöö tegemise käigus õpiti ja saadi uusi teadmisi kasutatud tehnoloogiate omandamiseks. Reaalse toimiva lahenduse ja praktiliste tegevuste läbi omandati kõige suuremad oskused, kuidas veebirakendust algusest lõpuni VikoPub näitel luua. Lisaks viidi läbi mais katsetus potentsiaalse kasutajaga, kellelt saadi lisaks autori ideedele ka otsese kasutaja mõtteid, kuidas valminud veebirakendust veel paremaks muuta.

Edasi tuuakse välja võimalused või planeeritavad tööd VikoPub veebirakenduse parendamiseks.

Admin kasutaja ja moderaatori põhilisteks ülesanneteks on VikoPub veebirakenduses haldamine ja seoses sellega võiks pärast sisselogimist nendele esimesena kuvada artiklite halduse vaate, kuna see on veebisüsteemi loomise peamiseks tegevuseks.

VikoPub veebirakenduses on hetkel võimalik kategooriaid lisada ja kustutada aga sarnaselt artiklitele võiks lisada olemasolevate kategooriate muutmise võimaluse, mis praegu ei ole realiseeritud.

Faili lisamine publikatsioonile toimub, vaid artikli muutmise juures aga selle võiks lisada ka kohe artikli lisamise vormile. Näiteks pdf faili lisades ja seda artikli eelvaates nimetusel vajutades, laaditakse pdf fail alla aga parema lahenduse jaoks võiks automaatselt kuvada pdf faili sisu brauseris uuel vahelehel.

Artiklidele on võimalik lisada failina ka pilte aga pilt kuvatakse artikli eelvaatesse väikselt ning uduselt ja kui vajutada pildi nimetuse peale siis laaditakse see seadmesse alla. Parema vaate jaoks tuleks pildi eelvaadet muuta ja täiendada. Artikli sisuosas võiks lingid olla aktiivsed.

Artiklite väljaandeliikide raadio, televisioon ja internet juurde peaks lisama ajaleht/ajakiri. Lisaks väljaandeliigile peaks saama ka lisada väljaande nimetust, näiteks kui valitakse raadio siis järgmiseks saab kirjutada väljaande nime „Kuku raadio". Väljaande nimi peaks olema realiseeritud sarnaselt nagu seotud inimesed või otsingusõnad. Artiklite muutmisel kustuta ja uuenda nupu kõrvale peaks lisama sulge nupu, mis pärast sulgemist läheb artiklite loetellu.

Otsimise tulemusena saadud loetelu peaks olema eraldatud, näiteks joonte või eristatavate värvidena. Otsingule peaks esimeseks panema võimaluse, mille järgi otsima hakatakse ja siis otsingu lahter. Praegu on eespool otsingusõna sisestamine aga kui otsingusõna on sissestatud ja hiljem valitakse, mille järgi otsitakse, siis automaatselt otsingulahter tühjendatakse. Otsingul peaks kuupäeva otsimisel olema võimalus otsida ainult kuu või ainult aasta või vahemiku, näiteks jaanuarist veebruarini, järgi. Kui otsingu kaudu ei tuvastatud artiklid siis peaks väljastama sõnumi, et „Otsingu järgi ei tuvastatud või ei leitud artiklit või artikleid".

Viimaseks võiks VikoPub veebirakenduses olla võimalus näiteks ühe konkreetse kategooria kolme viimase kuu artiklite loetelu väljavõte, mida saaks lihtsasti alla laadida või edasi saata.

KOKKUVÕTE

Lõputöö teema on „TalTech Virumaa kolledži publikatsioonide veebirakenduse loomine".

Käesolev lõputöö jaguneb kolmeks osaks. Esimeses peatükis tutvustatakse erinevaid veebirakenduse tehnoloogiaid, võrreldakse neid ja tuuakse välja loodava rakenduse jaoks tellimussoovina esitatud lahendus. Teises peatükis kirjeldatakse loodavat veebirakendust, selle nõudeid, loomist, struktuuri, koodinäiteid ja vaateid. Viimases sisupeatükis antakse ülevaade rakenduse tulemusest, hinnangust ja tuuakse välja võimalused edasiarendamiseks.

Käesoleva rakenduskõrgharidusõppe lõputöö eesmärgiks oli luua TalTech Virumaa Kolledžile veebirakendus nimetusega VikoPub, kuhu salvestada kolledžist ilmunud publikatsioonid.

Autori eesmärk luua veebirakendus, kuhu saaks TalTech Virumaa kolledžiga seonduvad publikatsioone koguda ja neid hallata, sai täidetud. VikoPub loomiseks kasutati MEAN virna ja JWT tehnoloogiat. MEAN virna kuuluvad MongoDB andmebaas, Express veebiraamistik, Angular kliendiosa raamistik ja Node.js veebiserveriosa. JWT autentimist kasutati turvalisuse tagamiseks. Veebirakendus koosneb kahest projektist: tagarakendus projektist nimega ArticleNodeExpress ja eesrakenduse projetist nimetusega ArticleAngular.

VikoPub veebirakendus on üleslaetud ja kättesaadavaks tehtud Heroku pilveteenuse platvormil. Valmis veebirakendus asub aadressil viko-pub.herokuapp.com. VikoPub võimaldab ühte kohta kokku koondada kõik kolledžiga seonduvad artiklid. Kõik funktsionaalsused ja nõuded, mis VikoPub arendamisel püstitati, on täidetud. Üldiselt võib jääda valminud tulemusega igati rahule. Veebirakendus töötab ning edasiarendamiseks esitatud ettepanekuid saab lisada või muuta, et VikoPub veel paremaks ja kasutajasõbralikumaks muuta. Probleemi lahendamise käigus on suurepärane võimalus praktiseerida õpitud oskusi ja see läbi luua reaalselt vajaminev ja kasulik lahendus.

Lisaks valmis VikoPub veebirakenduse loomise juhend (Lisa 2), mis koosneb taga- ja eesrakenduse õpetustest, tegevustest ja koodidest. Juhendit saab kasutada õppematerjalina, kui on vaja luua sama tehnoloogiaid kasutades sarnane veebilahendus.

SUMMARY

The topic of this thesis is "Creating a web application for TalTech Virumaa College publications".

This thesis is divided into three sections.

The first chapter introduces different web applications technologies, all of them are compared and in the end the one that is requested solution for the client is introduced. In the second chapter the created web application is described, its requirements, creation, structure, code examples and views. The last chapter gives an overview of the applications outcome, valuation and is brought out possibilities for future progress.

The purpose of this applied higher education thesis was to create TalTech Virumaa College web application named VikoPub, an application where all the college publications are issued.

The author's goal of creating a web application where publications related to Taltech Virumaa college could be collected and managed was fulfilled.

To create VikoPub it was used MEAN stack and JWT authentication. In MEAN stack are MongoDB database, Express web framework, Angular client framework and Node.js web server component. JWT authentication was used to provide security. Web application consists of two projects. Backend application project named ArticleNodeExpress and application project named ArticleAngular.

VikoPub web application is uploaded and available on Heroku cloud application platform. Ready web application is located on web address viko-pub.herokuapp.com. VikoPub lets gather together into one place all the articles that are associated with the college. All the functions and requirements that were set creating VikoPub were fulfilled.

In general, the results of the completed application are satisfactory. Web application is running and the future proposals can be added or changed to make VikoPub better and even more user friendly. Solving application problems is a great opportunity to practice learned skills and though that create a needed and useful solution.

Also, there are being prepared a manual to help with understanding VikoPub web application (Addition 2) that contains tutorials of the application, activities and codes. Manual can be used as a study material if there is a need to create web applications using the same technologies.

KASUTATUD ALLIKAD

[1] E. K. Instituut, „publikatsioon," Sõnaveeb, 2021. [Võrgumaterjal]. Available: https://sonaveeb.ee/search/unif/dlall/dsall/publikatsioon/1. [Kasutatud 09. 03. 2021].

[2] E. K. Instituut, „veebirakendus," Sõnaveeb, 2021. [Võrgumaterjal]. Available: https://sonaveeb.ee/search/unif/dlall/dsall/veebirakendus/1. [Kasutatud 09. 03. 2021].

[3] Cybernetica, „veebirakendus," Andmekaitse ja infoturbe leksikon, 2011 - 2021. [Võrgumaterjal]. Available: https://akit.cyber.ee/term/3965-veebirakendus. [Kasutatud 09. 03. 2021].

[4] T. P. D. Group, „PHP Manual," 1997-2021. [Võrgumaterjal]. Available: https://www.php.net/manual/en/. [Kasutatud 06. 03. 2021].

[5] M. Metshein, „PHP – Mis on PHP?," [Võrgumaterjal]. Available: https://www.metshein.com/unit/php-mis-php/. [Kasutatud 07. 03. 2021].

[6] S. Bhatia, „Best PHP Frameworks for Web Development," 2021 01. 08.. [Võrgumaterjal]. Available: https://hackr.io/blog/best-php-frameworks. [Kasutatud 2021 05. 22.].

[7] R. Data, „Java Introduction," W3Schools, 1999-2021. [Võrgumaterjal]. Available: https://www.w3schools.com/java/java_intro.asp. [Kasutatud 07. 03. 2021].

[8] V. Inc, „Web Applications," [Võrgumaterjal]. Available: https://spring.io/web-applications. [Kasutatud 2021 05. 22.].

[9] P. Javin, „Top 5 Programming languages for Web development in 2021," Medium, 13. 01. 2021. [Võrgumaterjal]. Available: https://medium.com/javarevisited/top-5-programming-languages-for-web-development-in-2021-f6fd4f564eb6. [Kasutatud 13. 02. 2021].

[10] R. Data, „Node.js Introduction," W3Schools, 1999-2021. [Võrgumaterjal]. Available: https://www.w3schools.com/nodejs/nodejs_intro.asp. [Kasutatud 08. 03. 2021].

[11] T. Point, „Node.js - Quick Guide," [Võrgumaterjal]. Available: https://www.tutorialspoint.com/nodejs/nodejs_quick_guide.htm. [Kasutatud 07. 03. 2021].

[12] Stackowerflow, „2020 Developer Survey - Most Popular Technologies," [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2020. [Kasutatud 05. 04. 2021].

[13] Section, „Introduction to hapi.js Framework," [Võrgumaterjal]. Available: https://www.section.io/engineering-education/introduction-to-hapi/. [Kasutatud 16. 03. 2021].

[14] „Koa," [Võrgumaterjal]. Available: https://koajs.com/. [Kasutatud 16. 03 2021].

[15] T. Point, „Node.js - Express Framework," [Võrgumaterjal]. Available: https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm. [Kasutatud 16. 03. 2021].

[16] Github, „hapijs/hapi," [Võrgumaterjal]. Available: https://github.com/hapijs/hapi. [Kasutatud 24. 04. 2021].

[17] Github, „koajs/koa," [Võrgumaterjal]. Available: https://github.com/koajs/koa. [Kasutatud 24. 04. 2021].

[18] Github, „expressjs/express," [Võrgumaterjal]. Available: https://github.com/expressjs/express. [Kasutatud 24. 04. 2021].

[19] T. Point, „ReactJS - Overview," [Võrgumaterjal]. Available: https://www.tutorialspoint.com/reactjs/reactjs_overview.htm. [Kasutatud 28. 03. 2021].

[20] Autocode, „What is Vue.js?," [Võrgumaterjal]. Available: https://vuejs.org/v2/guide/. [Kasutatud 28. 03. 2021].

[21] Google, „What is Angular?," Angular, 2010 - 2021. [Võrgumaterjal]. Available: https://angular.io/guide/what-is-angular. [Kasutatud 29. 03. 2021].

[22] A. Pattakos, „Angular vs React vs Vue 2021," aThemes, 25. 01. 2021. [Võrgumaterjal]. Available: https://athemes.com/guides/angular-vs-react-vs-vue/. [Kasutatud 29. 03. 2021.].

[23] G. Trends, „Vue.js, React ja Angular võrdlus," [Võrgumaterjal]. Available: https://trends.google.com/trends/explore?cat=31&date=2021-01-01%202021-12-31&q=Vue.js,React,Angular. [Kasutatud 01. 04. 2021].

[24] PostgreSQL, „About PostgreSQL," The PostgreSQL Global Development Group, 1996 - 2021. [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 02. 04. 2021].

[25] R. Data, „Introduction to MySQL," W3Schools, 1999 - 2021. [Võrgumaterjal]. Available: https://www.w3schools.com/mysql/mysql_intro.asp. [Kasutatud 02. 04. 2021].

[26] I. MongoDB, „What Is MongoDB?," [Võrgumaterjal]. Available: https://www.mongodb.com/what-is-mongodb. [Kasutatud 05. 03. 2021].

[27] T.T.T, „About mongoose," Stackowerflow, 2009. [Võrgumaterjal]. Available: https://stackoverflow.com/tags/mongoose/info. [Kasutatud 03. 04. 2021].

[28] MIT, „mongoose," [Võrgumaterjal]. Available: https://mongoosejs.com/. [Kasutatud 05. 03. 2021].

[29] M. Inc., „What is the MEAN Stack?," [Võrgumaterjal]. Available: https://www.mongodb.com/mean-stack. [Kasutatud 05. 04. 2021].

[30] BezKoder, „Angular 10 + MongoDB example with Node.js Express: CRUD App," [Võrgumaterjal]. Available: https://bezkoder.com/angular-10-mongodb-node-express/. [Kasutatud 10. 04. 2021].

[31] Cybernetica, „CORS," Andmekaitse ja infoturbe leksikon, [Võrgumaterjal]. Available: https://akit.cyber.ee/term/12532-cors. [Kasutatud 10. 04. 2021].

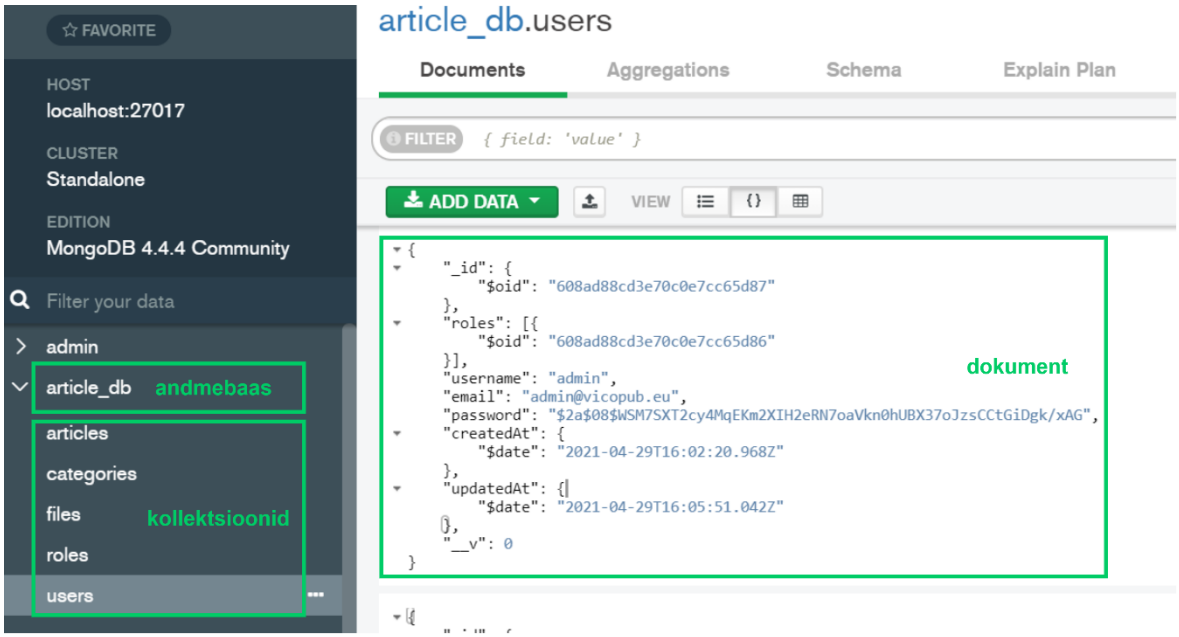
[32] JSON, „Introducing JSON," [Võrgumaterjal]. Available: http://www.json.org. [Kasutatud 05. 04. 2021].

[33] Auth0, „What is JSON Web Token?," [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 05. 03. 2021].

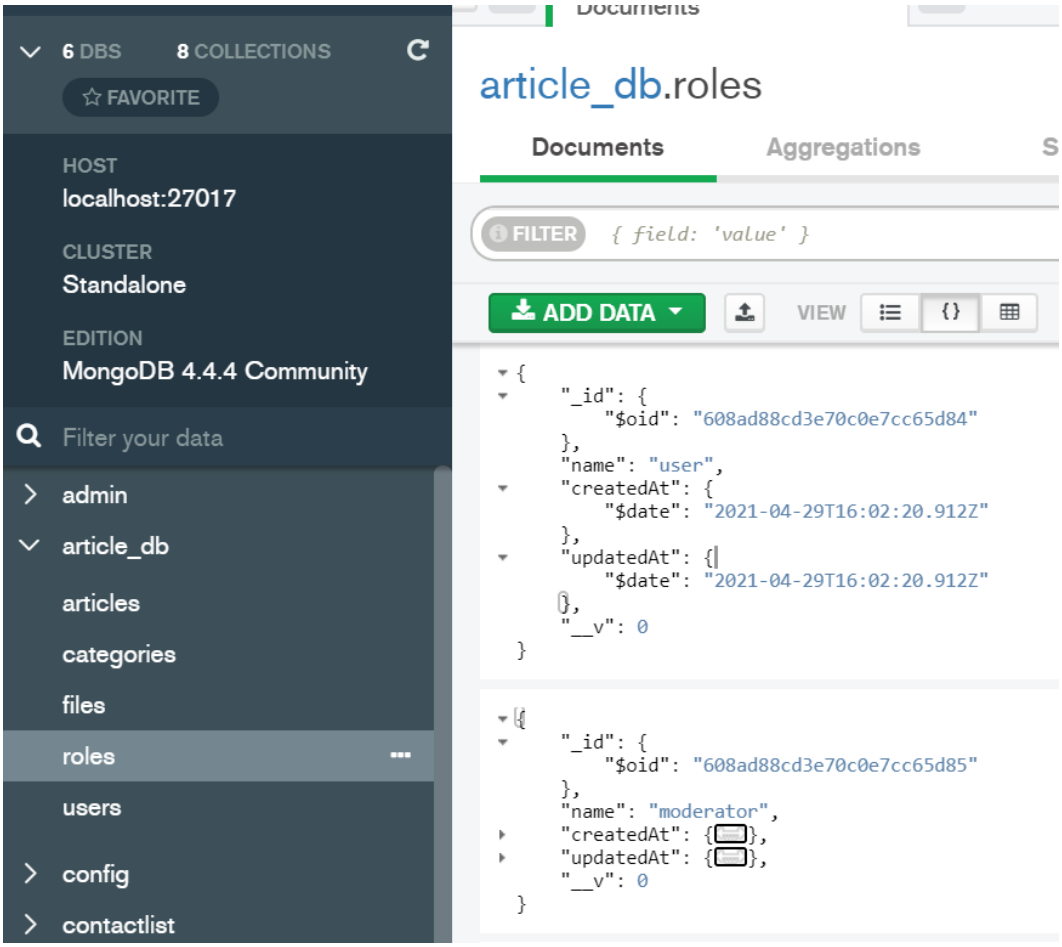
LISAD

Lisa 1 MongoDB andmebaasi kirjed

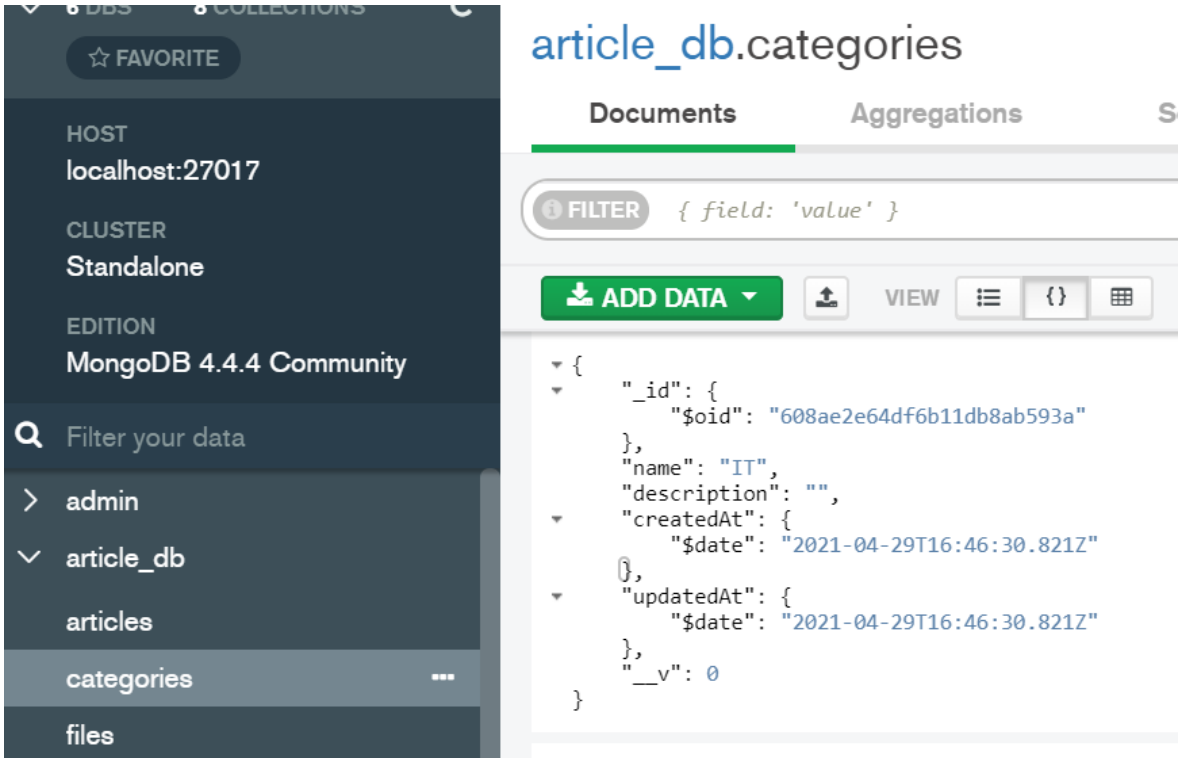
Kasutaja



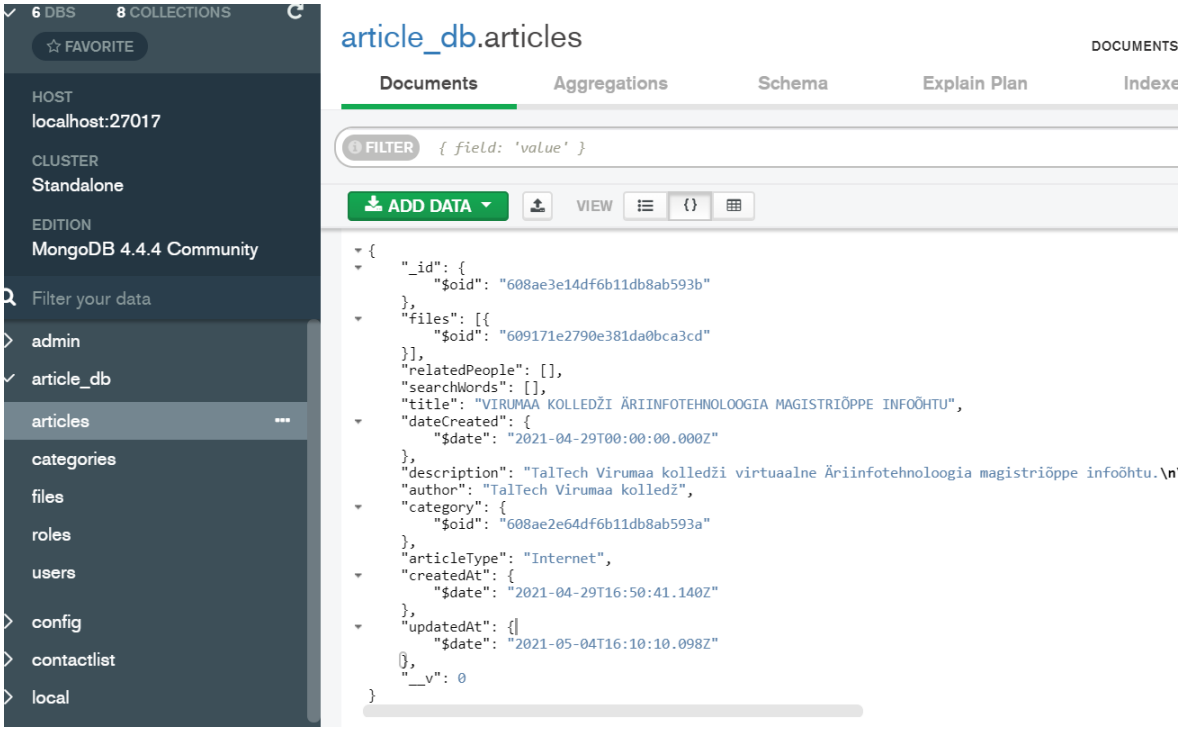
Rollid



Kategooria



Artikkel



Lisa 2 MEAN stack (MongoDB (mongoose), Express, Angular, Node.js) + JWT Authentication veebirakenduse loomise juhend VikoPub näitel

Õppematerjali VikoPub loomise juhendist on lisatud 3 esimest lehekülge. Orginaaljuhendis on kokku 62 lk.

Sissejuhatus

Töö eesmärgiks on TalTech Virumaa kolledžile luua veebirakendus nimetusega VikoPub, kuhu salvestada kolledžist ilmunud publikatsioonid. Veebirakenduse VikoPub loomisprotsessist peab valmima juhend, mis õpetab tegema sarnaseid veebirakendusi.

VikoPub veebirakenduse loomiseks on kasutatud bezkoder-i poolt loodud õpetusjuhiseid, mis asuvad https://bezkoder.com/ veebilehel.

VikoPub projekt on kombineeritud kokku kahest projektist:

1. ArticleNodeExpress (tagarakendus)
2. ArticleAngular (eesrakendus)

Juhend jaguneb järgmisteks osadeks, mis on ka sisukorraks:

Tarkvara installimine

Tagarakenduse projekti loomine

Vajalike moodulite installimine

Struktuur

Eesrakenduse projekti loomine

Vajalike komponentide, teenuste ja moodulite installimine

Projekti kaustade ja failide loomine (Angulari projekti loomisel kasutatakse component-ide ja service-te loomiseks ng s ja ng c käsklusi)

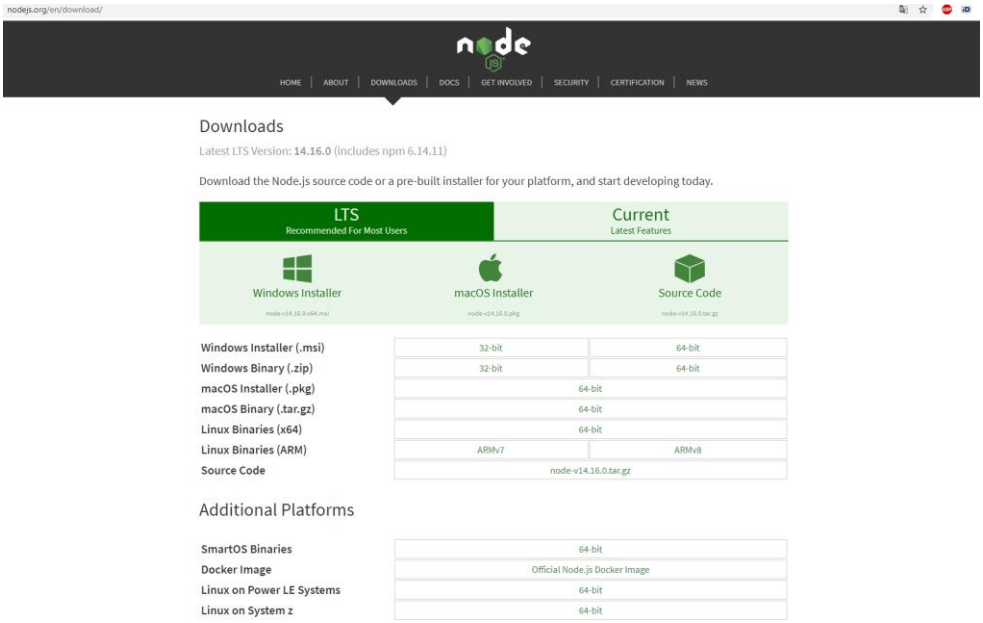
1. Tarkvara installimine

Veebirakenduse alustamiseks on kõigepealt vaja installida vajalik tarkvara.

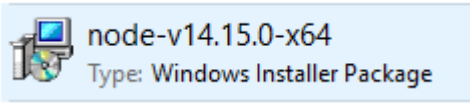
VikoPubi loomiseks on kasutatud IntelliJ IDEA programmi ja selle saab alla laadida nende ametlikult veebilehelt: https://www.jetbrains.com/idea/download

Teiseks on arvutisse vaja installida Node.js.

Node.js installeri alla laadimine käib Node.js ametlikult veebilehelt https://nodejs.org/en/download/.

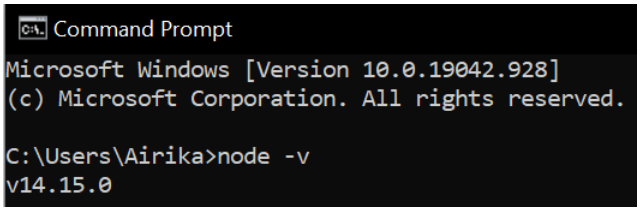


Valida enda süsteemile vastav versioon ja lae alla. Kui allalaadimine on lõppenud, käivita Node.js installer.



Nõustu tingimustega ja vajuta järgmine/next ning las installimise kaustaks jääb C:\Program Files\nodejs\. Vajuta järgmine/nest veel 3 korda ja viimaseks vajuta Install. Kui installimine on lõppenud vajuta Finish.

Edasi Node.js kinnitamiseks, käivita Käsuriba/Käsuviip/Command Prompt – cmd ja sisesta käsk node -v ja enter. Kui Node.js on edukalt installitud siis käslkus näitab, missugune Node.js versioon seadmes on.



2. Tagarakenduse ArticleNodeExpress projekti loomine

ArticleNodeExpress projekt hõlmab enda alla VikoPubi veebirakenduse MEAN stack tagarakenduse (backend) osa, kuhu peamiselt kuulub Express veebiserver, MongoDb andmebaasi ühendamine, milles kasutatakse mongoose mudeleid ja JWT autentimine.

2.1. Kausta loomine

Projekti kaustade loomiseks on järgnevalt 2 võimalust:

1) Ava IntelliJ IDEA programm. Kui IDEA on avatud siis vali File -> New -> Project. Kui esimeseks avaneb Welcome aken siis vali New Project -> JavaScript -> Next -> Pane enda projektile nimi " ArticleNodeExpress". Viimaseks vajuta Finish.

VÕI

2) cmd käskudena:
    a. mkdir ArticleNodeExpress (mkdir käsklus loob uue kausta/kataloogi)
    b. cd ArticleNodeExpress (cd käsklust kasutatakse kausta/kataloogi muutmiseks)

IntelliJ IDEA-s saab kasutada Terminali käskluste käivitamiseks.

Kui soovite võite cmd - s käsklusi edasi sisestada .

2.2. Vajalike moodulite installimine

ArticleNodeExpress projekt hõlmab enda alla VikoPubi veebirakenduse MEAN stack tagarakenduse (backend) osa, kuhu kuuluvad Node.js Express veebiserver, MongoDB andmebaas koos mongoose mudeliga, JWT autentimine.

Järgnevalt on täpsem juhend ArticleNodeExpress projekti loomise kohta

Projekti tuleb lisada npm (Node Package Manager)

npm install (npm installimine, mis lisab projekti package.json faili)

npm init (uue npm-i või olemasoleva lähtestamine või seadistamine)

```
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
```

...