TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Kethmar Salumets 120731IABB

SERVERI- JA KLIENDIPOOLNE VEEBISÜSTEEMIDE ARENDUS JAVASCRIPTI RAAMISTIKEGA KIFT.PLACE NÄITEL

Bakalaureusetöö

Juhendaja: Roger Kerse
Lektor

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kethmar Salumets

22.05.2017

Annotatsioon

Käesoleva töö eesmärgiks on võrrelda Javascripti serveri- ja kliendipoolseid veebisüsteemide raamistikke ning tulemuste põhjal arendada välja üritusteportaal Kift.place, selle halduspaneel ning rakendusliides.

Autor võtab vaatluse alla erinevad Javascripti veebisüsteemide arendamiseks mõeldud raamistikud, toob välja nende tugevused ning nõrkused ja analüüsib nende sobivust käesoleva bakalaureusetöö raames arendatava teenuse ehitamiseks. Lisaks tutvustatakse teisi tähtsaid tehnoloogiaid, mida nimetatud teenuse ehitamiseks vaja.

Töö tulemusena valmis avalikuks kasutamiseks mõeldud üritusteportaal Kift.place, selle halduspaneel ning rakendusliides, mida saavad kasutada potentsiaalsed koostööpartnerid erinevate teenuste loomiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 58 leheküljel, 5 peatükki, 22 joonist ning 6 tabelit.

Abstract

The aim of this thesis is to examine different Javascript client- and server-side web frameworks and based on the examination, choose the most suitable ones for developing a website for events called Kift.place, its content management system and application programming interface.

The author takes under scrutiny a selection of Javascript frameworks meant for building web systems, brings out the more important aspects of them and tries to find out which one suits best for building the aforementioned service. Additionally, other important technologies used for building Kift.place system are introduced.

The result of this thesis is public website Kift.place which can be used to find events in Tallinn, its content management system meant for the owners of the project and an application programming interface for potential partners.

The thesis is written in Estonian and contains 58 pages of text, 5 chapters, 22 figures and 6 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface* Rakendusliides ehk programmiliides on reeglite kogum olemasoleva valmisprogrammiga suhtlemiseks. [1]
Token	Ligipääsu *token* on objekt, mis sisaldab endas kasutaja kohta käivat informatsiooni ning tema õigusi. [2]
Data-binding	Tehnika, mis seob kasutajaliideses oleva informatsiooni vastava mudelis oleva informatsiooniga. [3]
Middleware	*Middleware'*d on Node.js'is kasutatavad funktsioonid, mis saavad mõjutada ning töödelda sissetulevate ning väljaminevate päringute objekte. [4]
MVW	*Model-View-Whatever* Angular JS poolt kasutusele võetud termin, mille kohaselt on tarkvaral olemas mudel, vaade ning kolmas komponent, mis arendajale endale kõige paremini sobib. [5]
MVC	*Model-View-Controller* Tarkvaraarenduses tuntud muster, mille kohaselt on rakenduse arendusel olemas mudel, mis tähistab informatsiooni, vaade ehk kasutajale nähtav osa ning kontroller, mis sündmustele vastab. [6]
Callback	*Callback* funktsioon on Javascriptis tuntud kui käivitatav kood, mida saab anda teistele funktsioonidele argumentidena kaasa. Argument käivitatakse soovitud ajal. [7]
ORM	*Object-Relation mapping* Programmeerimises kasutatav tehnika, kus relatsiooniliste andmebaaside kirjed muundatakse objekt-orienteeritud kujule ümber. [8]
DOM	*Document Object Model* Rakendusliides HTML'i ja XML'i kirjutamiseks, mis defineerib dokumendi struktuuri. [9]

Sisukord

Jooniste nimekiri

Tabelite nimekiri

1. Sissejuhatus

Võimalusi tänapäeval veebisüsteemide ehitamiseks on erinevaid. Traditsioonilised programmeerimiskeeled nagu Python, Java ja PHP pakuvad raamistikke veebilehekülgede loomiseks, samas viimastel aastatel on antud valdkonnas kasvanud ka Javascripti populaarsus. Keel, mida algselt kasutati eelkõige kasutajaliideste ehitamiseks, on tänapäeval kasutusel ka taustsüsteemide realiseerimisel. [10] Oma universaalsuse tõttu on nimetatud programmeerimiskeel tänapäeval üks populaarsemaid, mistõttu leidub raamistike ning teeke ühesuguste probleemide lahendamiseks palju. [11] Sellest tulenevalt on kvaliteetse lõpp-produkti realiseerimiseks vajalik vaadelda ja analüüsida erinevaid raamistikke ning valida välja just teenuse ja arendaja huvidest lähtuvalt parim.

Bakalaureusetöö eesmärgiks on vaadelda valikut kliendi- ja serveripoolseid Javascripti raamistikke ning neid võrrelda. Vastavalt tulemustele valib autor välja sobivad raamistikud, mille abil luua avalikuks kasutamiseks mõeldud üritusteportaal nimega Kift.place koos kolmandate osapoolte poolt kasutatava rakendusliidese ja halduspaneeliga.

Käesoleva kirjatöö teoreetilises osas võtab autor vaatluse alla kolm serveripoolset ning kolm kliendipoolset raamistikku, mis on kirjutatud Javascriptis.

Bakalaureusetöö praktilises osas tutvustab autor loodavat teenust, selle nõudeid ning analüüsib käesoleva töö esimeses osas vaadeldud raamistikke. Lisaks toob autor välja palju teisi teenusega seotud tehnoloogiaid, mis lõpp-produkti loomisel vajalikud olid.

Osas „Valminud teenus“ esitleb autor valminud teenust ja selle võimalusi ning toob välja puudujääke.

2. Veebiarendusvõimalused Javascriptis tänapäeval

Käesoleva peatüki raames vaatleb autor erinevaid raamistikke veebisüsteemide ehitamiseks Javascriptis.

2.1 Taust ja probleem

Võimalusi tänapäeval veebisüsteemide ehitamiseks on palju. Erinevad traditsioonilised programmeerimiskeeled nagu Java, PHP ning Python on pikka aega pakkunud raamistikke veebilehekülgede loomiseks, kuid viimastel aastatel on enim populaarsust hakanud koguma Javascript – keel, mis suudab pakkuda kasutusvõimalusi nii serveri kui ka kliendi poolel. [10, 12]

Javascripti suurimaks eeliseks on selle universaalsus. Kasutades üht programmeerimiskeelt, on arendajal võimalik kirjutada koodi nii serveri kui ka kliendi poolele, pakkudes kergesti hoomatavat terviklahenduste loomisvõimalust nii tagasüsteemi kui ka eessüsteemi arendajatele. Olles sobilik valik nii suurele osale tarkvara kirjutajatest, on Javascriptist saanud üks maailma enimkasutatud programmeerimiskeeli. [11] Samas leiab käesoleva töö autor, et populaarsusega kaasneb ka palju probleeme.

Suure kasutajaskonna ning nende erinevate soovide tõttu on paljud arendajad panustanud Javascripti raamistike ning moodulite loomisesse. Ühelt poolt teeb see tarkvara kirjutavate inimeste elu lihtsaks, sest valikuid probleemide lahendamiseks on palju, teisalt nõuab kvaliteetse lõpp-produkti loomine rohkem läbimõeldud otsuseid – milliseid lahendusi kasutada ning mis eesmärgil? Node Package Manager(edaspidi NPM), tööriist Javascripti teekide ning raamistike haldamiseks, koondab kokku enam kui 465 000 Javascripti moodulit. [13] Nii suure hulga võimaluste seast õigete leidmine ning nende nõuete järgimine on raske, kuid samas tähtis protsess.

Käesoleva bakalaureusetöö raames uurib autor valikut populaarsemaid veebisüsteemide loomiseks kasutatavaid Javascripti raamistikke nii kliendi- kui ka serveri poolel ning

valmistab väljavalitud raamistike põhjal avalikult kasutamiseks mõeldud ürituste portaali nimega Kift.place ja selle rakendusliidese ning halduspaneeli.

2.2 Populaarsemad serveripoolsed Javascripti raamistikud

Käesolevates alapeatükkides võtab autor vaatluse alla valiku Node.js, serveripoolse Javascripti, veebiraamistikke ning toob välja nende tunnusjooni, mida võib veebiarenduse serveri pooles tähtsaks pidada.

2.2.1 Express

Express on Node.js veebisüsteemide loomiseks mõeldud raamistik, mis pakub arendajale kõige tavalisemaid funktsioone, mida veebiserveri ehitusel vaja, nende hulgas HTTP päringute käsitlemine, mallide koostamine ning esitamine vastavalt nõuetele, kuid samas ka võimalust ära kasutada Node.js *middleware*'sid sissetulevate päringute manipuleerimiseks. [14]

Sissetulevate päringute käsitlemise lihtsuse tõttu kasutatakse Expressi palju REST teenuste loomiseks. Seepärast vaadeldakse antud raamistiku ka kui Node.js API ehitamiseks mõeldud tööriista. [15]

Suur Expressi tunnus on valikuvabadus. Antud raamistik ei sea piiranguid tehnoloogiate ning projekti ülesehituse koha pealt. Arendaja saab valida, mis mallide koostamistehnoloogiat, andmebaasi ja süsteemi arhitektuurimustrit ta soovib jälgida. [16] See seab eelise paljude teiste sarnaste raamistike ees, sest ei survesta programmeerijat jälgima kindlaid juhiseid ning laseb süsteemi ehitada vastavalt sobivusele.

Olles üks enimkasutatavamaid mooduleid Javascriptis, on Expressi kasutajaskond võrreldes oma konkurentidega suur ning see teeb arendustöös vajamineva dokumentatsiooni, õpetuste ning näidiste leidmise kergeks. [17] 2017. aasta aprillikuu jooksul tõmmati NPM'ist antud raamistiku alla 10 925 693 korda, mis näitab Expressi stabiilsust ning kasutusvalmidust. [18]

Oma funktsionaalsuse ning populaarsuse tõttu on Express ennast tõestanud raamistik veebisüsteemide ehitamiseks, jättes kasutajale võimaluse otsustada rakendatavate tehnoloogiate ning arhitektuurimustrite osas.

2.2.2 Hapi

Hapi on Node.js raamistik, mis pakub arendajale vajalikke funktsioone oma veebisüsteemi loomiseks Javascriptis. Sarnaselt Expressile, saab Hapi abil töödelda serverisse tulevaid päringuid ning genereerida malle, kuid eriliseks teeb selle sisseehitatud konfigureerimisvõimaluste rohkus. [19]

Kõnealune raamistik on mõeldud eelkõige modulaarsete süsteemide kirjutamiseks, kus on vähem programmeerimist ning rohkem konfigureerimist. Iga päringu lõpp-punkti tuleb vastavalt seadistada. Kohustulik on määrata ära HTTP meetod ning teekond päringuni, kuid samas saab muuhulgas määrata ka päringute sisutüüpe, *caching* parameetreid ja päiseid. Selline võimalusterohkus on abiks arendajale, kes soovib omada rohkem kontrolli oma süsteemi üle, kasutamata lisasid. [20]

Hapi eripäraks on ka sisseehitatud sisu valideerimine ja autoriseerimismeetodite olemasolu. Antud funktsionaalsus on tähtis veebisüsteemides, kus on palju kasutaja poolt täidetavaid vorme ning kasutajahaldust. Seetõttu sobib kõnealune hästi haldussüsteemide ehitamiseks. [21]

Lisaks nimetatule on Hapi teistest raamistikest erinev pistikprogrammide kirjutamise võimaluse tõttu. Neid saab kasutada individuaalsete lõpp-punktide haldamiseks ning lisafunktsionaalsuse läbiviimiseks. Antud võimekus sarnaneb Node.js *middleware'*le, kuid pakub rohkem modulaarsust, sest kutsutakse välja vaid soovitud kohtades. [22]

Kasutajatehulga ning kommuuni suuruse poolest jääb Hapi selgelt Expressile alla – antud moodulit on alla laetud 2017. aprillis 279 450 korda. [21] Sellegipoolest leiab käesoleva töö autor, et tegu on piisavalt suure arvu inimestega pidamaks Hapit tõsiseltvõetavaks raamistikuks.

Arendajale, kes soovib omada palju kontrolli sissetulevate päringute üle ning samas kasutada sisseehitatud struktuurile vastavat autoriseerimissüsteemi, on selle kirjatöö autori silmis Hapi sobilik valik. Lisaks pakub antud raamistik võimalust ehitada lisasid, mida kasutatakse kindlatele lõpp-punktidele minevate päringute ajal.

2.2.3 Sails.js

Sails.js on veebisüsteemide raamistik, mis ehitatud kasutades ära Expressi ja Javascriptis reaalajas suhtluse pidamiseks loodud Socket.IO raamistikku. Sails.js rakendab MVC tarkvaramustrit ning sarnaneb oma ülesehituselt Ruby on Rails raamistikule. [23, 24]

Sails erineb teistest mainitud raamistikest kõige rohkem oma nõuete pärast. Antud raamistiku peale ehitatud rakendused peavad rangelt jälgima MVC tarkvaramustrit. See tähendab, et arendaja jaoks jääb valikuvõimalusi vähemaks, kuid samas on rohkem ettekirjutisi, kuidas asju õigesti teha. Sails rakendustes on kaustade ning failide sisu struktuur kindlalt ära määratletud.

Kõnealuse raamistiku järgmiseks tunnuseks on andmebaasiga suhtlemiseks kasutusele võetud ORM nimega Waterline. See tähendab, et arendaja võib kasutada mõnda toetatud andmebaasisüsteemi ning suhelda sellega läbi Sails'is rakendatava andmetöötluskihi. [24] Selline võimalus on positiivne, kui arendaja on juba tuttav Waterline'ga või plaanib Sails rakendust kasutada pikemaajaliselt.

Üheks suureks tugevuseks Sails raamistikus on selle võime pidada reaalajas suhtlust. Antud raamistik on ehitatud ära kasutades Socket.IO moodulit, mis võimaldab kõikidel Sails rakendustel soovi korral saata päringuid ning ning neid vastu võtta reaalajas nii, et klient ei pea lehte taaslaadima muudatuste nägemiseks. [25] Selline omadus on kasulik näiteks veebisüsteemidena üles ehitatud jututubadele.

Sails'i kasutajate hulk ei ole NPM põhjal suur. 2017. aasta aprillis tõmmati seda alla 55 952 korda. [26] Sellest järeldatuna ei ole antud raamistiku populaarsus võrreldes oma konkurentidega suur. See tähendab, et nõu küsimine ning näitematerjalide leidmine ei pruugi olla alati saadaval.

Sails on raamistik, mis võrreldes Expressi ja Hapiga on struktureeritum ning pakub rohkem algseid võimalusi. Kasutades ära MVC tarkvaramustrit, nõuab see tarkvaraarendust kindlaid reegleid järgides. Samas pakub Sails reaalajas suhtlust ning üht kindlat viisi andmebaasidega suhtlemiseks, mis käesoleva töö autori silmis võib vastavalt arendaja soovidele olla tugev argument raamistiku kasutuselevõtmiseks.

2.3 Populaarsemad kliendipoolsed Javascripti raamistikud

Järgnevates alapeatükkides toob autor välja kolm Javascripti raamistikku, mis on mõeldud üheleheküljeliste kliendipoolsete veebirakenduste loomiseks.

2.3.1 Angular JS

Angular JS on kliendipoolne Javascripti raamistik, mille arendamist juhib Google. Antud moodul on mõeldud staatiliste HTML lehekülgede dünaamiliseks muutmiseks ning see järgib MVW tarkvaramustrit. [27]

Angular töötleb HTML'i läbi kahepoolse *data-bind*'ingu. See tähendab, et kui kliendi pool muutub miskit, saadetakse informatsioon automaatselt ka Angulari mudelitesse, samas kui midagi muutub mudelites, saadetakse informatsioon koheselt ka kliendi vaatesse. Kogu seda protsessi juhib mudeli ja vaate vahele jääv kontroller. [28] Selline funktsionaalsus vabastab arendaja DOM'i manipuleerimise protsessidest ning laseb keskenduda rakenduse funktsionaalsusele.

Angular on suur raamistik, mis hõlmab endas palju erinevaid mooduleid. Viimaseid kasutatakse kindlate ülesannete täitmiseks. Näiteks on igal Angulari rakendusel kasutusel baasmoodul, mis pakub rakenduse baasfunktsioone, kuid samas on võimalik vormide valideerimiseks kasutada vormide moodulit, suunamiseks suunamismoodulit ning HTTP päringute saatmiseks HTTP moodulit. Moodulite positiivseks küljeks on see, et neid saab juurde arendada ka kolmandate osapoolte poolt – see laiendab Angulari võimalusi märgatavalt. [29]

Eelmainitule lisaks on Angulari eripäraks ka direktiivid– funktsionaalsus, läbi mille arendaja saab luua enda HTML elemente. Igale direktiivile on võimalik seada kindel funktsionaalsus, struktuur ning parameetrid, mida kasutatakse kontrolleri ja mudelitega suhtlemisel. [30] See tähendab, et arendaja saab koostada dokumendi vaated, mis on kompaktsemad ning arusaadavamad, vähendades koodiridade arvu.

Angular on populaarne raamistik – NPM'is on seda 2017. aasta aprillikuu jooksul alla tõmmatud 781 453 korda. [31] Lisaks on antud raamistikul Google tugi, mis lubab eeldada korraliku dokumentatsiooni ning õpetuste leidmist internetis.

Angular on täisfunktsionaalne raamistik, mida toetab üks maailma suurimaid infotehnoloogia ettevõtteid. Jälgides MVW mudelit, seab käesolev raamistik arendajale konkreetseid nõudeid. Samuti tuleb Angulariga kaasa sissepakitud mooduleid, mida arendajal ei pruugi vaja minna. Käesoleva töö autor leiab, et Angular sobib arendajale, kellel on kogemusi MVC raamistikega ning kelle projektid kasutavad ära raamistiku erinevaid sisseehitatud mooduleid.

2.3.2 React JS

React JS on Javascriptil põhinev kliendipoolne Javascripti raamistik, mille eestvedajaks on Facebook. Reacti eesmärk olla tööriist suurte, üheleheküljeliste veebirakenduste loomisel, pakkudes kiiret, lihtsat ning skaleeruvat lahendust. [32]

Reacti peamiseks tunnuseks on selle võime manipuleerida DOM'i võimalikult väikese pingutusega. Selleks kasutatakse *virtual DOM*'i ehk kiiret mälusisest dokumendi struktuuri abstraktsiooni. Kui veebileheküljel midagi muutub, koostatakse ka uus *virtual DOM*, mille põhjal arvutatakse algoritmide järgi välja erinevus uue ja vana versiooni vahel. Pärast arvutusi uuendatakse veebilehekülje DOM ära, muutes vaid neid komponente, mis vaja. Reacti manipuleerimiskiirus tulebki faktist, et dokumendis muudetakse vaid need osad ära, mida reaalselt vaja on. [33]

Teine tähtis iseloomujoon on komponentidel põhinev struktuur. React eeldab rakenduse ehitamist blokkidena. See tähendab, et arendaja peab oma vaated väikesteks, iseseisvateks tükkideks jagama, millel igaühel on oma funktsionaalsus. Erinevaid komponente kombineerides tekib üks tervik. [34]

Reacti üheks omaduseks on ka ühesuunaline andmete liikumine. Informatsiooni muutmiseks komponendis tuleb need talle edastada läbi atribuutide. Komponendil puudub võime neid atribuute iseseisvalt muuta, kuid on võimalik välja kutsuda *callback* meetodid, mis tegelevad infokildude manipulatsiooniga. See teeb komponendid iseseisvamaks ning lubab neid kasutada erinevates kohtades vastavalt vajadusele. [33, 34]

Kuna React on eelkõige vaadete manipuleerimiseks mõeldud, ei ole antud raamistikku palju mooduleid sisse ehitatud, mis annab arendajale vabaduse valida, milliseid lisasid

kasutada. Näiteks HTTP päringute ning olekute haldamiseks on programmeerija sunnitud kaasama kolmandate osapoolte mooduleid. [35, 36]

Kõnealune raamistik on kasutajate arvu poolest väga populaarne. 2017. aasta aprillikuus tõmmati Reacti NPM'is alla 4 098 181 korda, millest võib eeldada, et Reacti kommuun ning materjalide kättesaadavus on väga suur. [37]

React sobib arendajale, kelle rakenduses on palju vaadete manipuleerimist ning info muutumist. Samuti pakub React uuenduslikku komponentidepõhist vaadete struktuuri, mis pikas perspektiivis võib olla suure projekti puhul jätkusuutlik valik. Samas peab arendaja arvestama, et React on mõeldud eelkõige vaadete töötlemiseks ning lisamooduleid, mida projekti kaasata, on palju.

2.3.3 Vue JS

Vue JS on Javascripti raamistik kasutajaliideste ehitamiseks. Selle ehitamisel on rõhutud vastuvõtlikkusele teiste moodulite ning raamistike suhtes. Erinevalt eelnevalt mainitud raamistikest, ei ole Vue loodud suurettevõtete poolt, kuid nende sponsorite hulka kuuluvad tuntud firmad nagu JSFiddle, Laravel ning Stdlib. [38, 39]

Sarnaselt Reactile on Vue's kasutusel komponentidel põhinev arhitektuur. See tähendab, et arendaja ehitab väikeseid blokke, mis kokkupanduna moodustavad ühe terviku. Vue komponent on oma olemuselt Vue objekti instants, millel on kindlaks määratud seaded. Neile seadetele saab kasutaja määrata erinevaid väärtusi nagu komponendi malli struktuur, võimalikud atribuudid ja selle infokillud.

Kõnealuse raamistiku teiseks tunnusjooneks on deklaratiivne renderdamine DOM'is. See tähendab, et muutes informatsiooni Vue komponendis muutub see ka DOM'is ning vastupidi. Sarnaselt Reactile, on siin kasutusel *virtual DOM*, mis kindlate arvutuste põhjal suudab leida kõige minimaalsema muudatuste hulga, mida HTML dokumendi struktuuris vaja teha on. [40] Selline käitumismuster on tervitatav ühelehelistes veebirakendustes, kus kasutaja eeldab kiiret tagasisidet oma tegevusele.

Vue ehitamisel on rõhku pandud vastuvõtlikkusele, mis väljendub väheses sisseehitatud moodulite hulgas. Arendaja saab ise valida, milliseid lahendusi ta soovib kasutada, et tekitada näiteks komponentidevahelisi suunamisi ja HTTP päringuid. See jätab valikuvabaduse tarkvara loojale, mis tähendab ka rohkem kontrolli rakenduse üle.

Üheks peamiseks Vue JS eeliseks konkurentide ees on ühe faili sisse ehitatud komponendid. Kogu komponendi Javascript, CSS ning HTML kood on võimalik panna ühte faili, mis vähendab nende hulka süsteemis ning annab parema ülevaate arendajale. Selline funktsionaalsus on tervitatav nii väikestes kui ka suurtes projektides, sest hoiab tarkvara failisüsteemi kompaktsena. [40]

2017. aasta aprillikuus laeti antud raamistikku NPM'ist alla 492 097 korda, millest võib järeldada, et Vue'l on piisavalt aktiivseid kasutajaid, kes antud lahendust oma projektides kasutaksid. [41]

Vue JS on kasutajaliideste ehitamiseks mõeldud raamistik, mis annab arendajale vabad käed moodulite valimisel ning pakub lõppkasutajale kiiret tagasisidet otsuste tegemisel. Eriliseks teeb Vue selle ühe faili komponentide võimalikkus, mis aitab projekti hoida kompaktse ja lihtsasti jälgitavana.

3. Teenuse loomine

Antud peatükis kirjeldab autor Javascripti raamistikel ehitatavat teenust, selle nõudeid ning uurib peatükis 1 välja toodud tehnoloogiate sobivust projekti ehitamiseks. Analüüsi põhjal valitakse välja konkreetne serveri- ja kliendipoolne raamistik, mida teenuse arenduses kasutatakse. Lisaks tehakse ülevaade teistest tehnoloogiatest ning moodulitest, mida eesmärgi saavutamiseks rakendatakse.

3.1 Teenuse kirjeldus

Käesoleva bakalaureusetöö raames valmiv teenus kannab nimetust Kift.place ning selle eesmärk on koondada ühele veebileheküljele erinevad Tallinnas toimuvad üritused, pakkudes seejuures intuitiivset kasutajaliidest ning ürituste kategoriseerimist. Lisaks veebileheküljele loob autor halduspaneeli, rakendusliidese ning automaatsüsteemi, mis suudab Facebookist vastavalt kindlatele parameetritele uusi üritusi süsteemi sisestada.

Teenus jaguneb loomeprotsessis kolmeks – rakendusliidese loomine, halduspaneel ning avalik veebilehekülg.

Halduspaaneli teenus on käesolevas projektis vajalik selleks, et ettevõtmisega seotud inimestel oleks ülevaade ning kontroll süsteemis olevate ürituste üle. Võimalik peab olema hallata erinevaid objekte nagu asukohad ja kategooriad, mille põhjal üritusi filtreeritakse. Lisaks on vajalik ürituste info muutmist - sealhulgas nime, kirjelduse, kuupäeva ning tunnuspildi - võimaldavaid funktsioone.

Projekti avalik pool, kõigile kasutatav veebilehekülg, koondab endas halduspaneelis avalikuks tehtud üritused. Eesmärk on pakkuda lõppkasutajale intuitiivset kogemust Tallinnas toimuvate ürituste leidmiseks võimalikult vähese vaevaga. Leheküljel on kategooriad, mis kuvavad vastava kategooria üritusi ning tutvustusleht, kus on lehekülje loojate kirjeldus ning kontaktandmed. Avalik veebilehekülg on kasutatav mobiilis ja arvutis.

Rakendusliides on antud projekti juures tähtis selleks, et avalik veebilehekülg ning halduspaneel teenust kasutada saaksid. Vastavalt vajadusele ning õigustele, saab antud funktsioonaalsuse abil läbi viia erinevaid toiminguid ja pärida andmeid. Nimetatud funktsionaalsust on võimalik kasutada ka potentsiaalsetel koostööpartneritel.

3.2 Teenuse nõuded

Teenuse iga alamosa omab erinevaid nõudeid, mis on välja toodud järgnevates alapeatükkides.

3.2.1 Halduspaneeli nõuded

Halduspaneelile esitatud nõuded on järgmised:

- Teenuse kasutamiseks on vaja kasutajakontot
- Teenuse kasutajaid saavad lisada vaid teenuse loojad
- Teenusel on kahte tüüpi kasutajad
    - Tavakasutaja
    - Admin kasutaja
- Teenuse kasutamiseks peab kasutaja sisestama enda e-posti ning parooli.
- Teenus peab valede andmete sisestamisel kasutajat sellest teavitama.
- Tavakasutaja peab saama näha enda üritusi ning neid redigeerida, kustutada.
- Admin kasutaja peab saama näha kõikide kasutajade üritusi ning neid redigeerida, kustutada.
- Admin kasutaja peab saama näha kõiki süsteemis olevaid kategooriaid ning neid redigeerida, kustutada.
- Admin kasutaja peab saama näha kõiki süsteemis olevaid asukohtasid ning neid redigeerida, kustutada.
- Admin kasutaja peab saama lisada uusi üritusi, kategooriaid ning asukohtasid.
- Asukoha loomisel ning redigeerimisel peab olema võimalik manipuleerida järgnevat informatsiooni:
    - Asukoha nimetus
    - Asukoha aadress
    - Asukoha eraldusnumber, mis vastab asukoha Facebooki eraldusnumbrile
    - Asukohaga seotud kategooriad

- Kategooriate loomisel ning redigeerimisel peab olema võimalik manipuleerida järgnevat informatsiooni:
    - Kategooria nimetus
    - Kategooria erilisus
- Ürituse loomisel ning redigeerimisel peab olema võimalik manipuleerida järgnevat informatsiooni
    - Nimi
    - Asukoht
    - Toimumise aadress
    - Kategooriad
    - Algusaeg
    - Lõppaeg
    - Kirjeldus
    - Tunnuspilt
- Üritusi peab saama filtreerida kategooria, asukoha ning avalikustamise järgi.
- Halduspaneel peab automaatselt iga 30 minuti tagant uuendama Facebook'ist andmeid
- Halduspaneel peab automaatselt saatma igale admin kasutajale kord nädalas e-kirja vastutusteatega.

3.2.2 Avaliku veebilehekülje nõuded

Teenuse avaliku veebilehekülje nõuded on järgmised:

- Teenus peab olema kättesaadav aadressil www.kift.place
- Teenuse informatsioon peab olema vastavuses halduspaneeli informatsiooniga
- Teenuse leheküljel peab nähtaval olema kategooriate menüü
- Teenuse leheküljel peab nähtaval olema sektsioon loojate tutvustusega
- Teenuse lehekülje kategooriate lehekülg peab sisaldama antud kategooriaga seotud üritusi
- Ürituse vaatel peab selgelt olema välja toodud:
    - Toimumiskoht
    - Toimumisaeg
    - Pealkiri

    - Kirjeldus
    - Tunnuspilt, kui on olemas
- Teenuse lehekülg peab olema responsiivne- kasutatav arvutis ja mobiilis.

3.2.3 Rakenduskihi nõuded

API teenuse nõuded on järgmised:

- Teenus omab avalikku ja privaatset poolt
- Avalik pool omab lõpp-punkti, mis tagastab avalikuks näitamiseks sobivad:
    - Üritused
    - Asukohad
    - Kategooriad
- Privaatne pool on kasutatav vaid:
    - Halduspaneeli sisse loginud kasutaja token'i põhjal
    - Kindlaks määratud salajase token'i põhjal
- Ebakorrektse või puuduva tokeni puhul tagastatakse veateade
- Privaatne pool peab pakkuma ürituste, kategooriate ja asukohtade lisamist, muutmist ja kustutamist.
- Privaatne pool peab pakkuma järgnevaid lõpp-punkte:
    - Kõikide ürituste lugemine
    - Individuaalse ürituse lugemine
    - Individuaalse ürituse muutmine ja kustutamine
    - Kõikide asukohtade lugemine
    - Individuaalse asukoha lugemine
    - Individuaalse asukoha muutmine ja kustutamine
    - Kõikide kategooriate lugemine
    - Individuaalse kategooria lugemine
    - Individuaalse kategooria muutmine ja kustutamine
    - Kõikide kasutajate lugemine

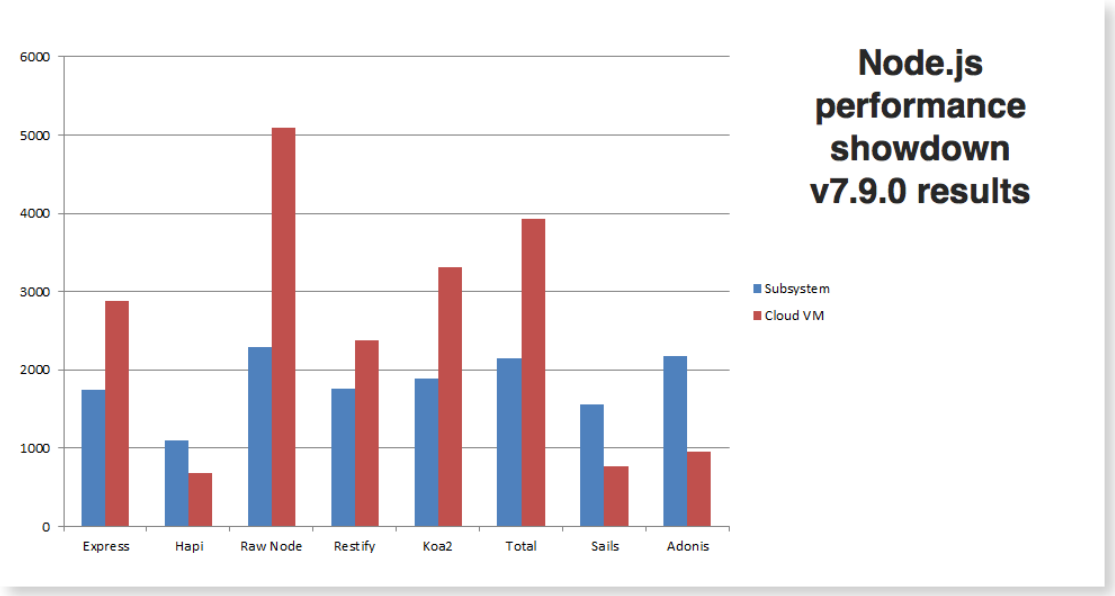
3.3 Serveripoolse veebisüsteemi raamistiku valimine

Järgnevates alampunktides võrdleb autor Express, Hapi ja Sails raamistikke ning valib välja neist sobivaima lähtuvalt projekti nõuetest. Eelkõige peab autor tähtsaks raamistiku jõudlust, kommuuni suurust ning valikuvabadust otsuste tegemisel.

3.3.1 Jõudlus

Antud bakalaureusetöö autori silmis on jõudluse võrdlemine raamistikes tähtis, sest teenuse populaarsemaks muutumise puhul peab server olema suuteline vastata võimalikult paljudele päringutele. Jõudluse võrdluseks kasutab autor Raygun.com poolt läbiviidud Node.JS veebisüsteemide ehitamiseks mõeldud raamistike jõudlustesti. [42]

Testimine seisnes iga raamistiku abil serveri ülespanemises Ubuntu alamsüsteemi ning Digital Ocean virutaalsüsteemi ja serverisse saadetud päringutele vastuse saamise arvus ühes sekundis. Iga raamistiku vastavasse serverisse saadeti korraga 100 GET päringut kuni 50 000 päringu täitumiseni või 20 sekundi möödumiseni. Testi tulemused on näha joonisel (vt Joonis 1).



Joonis 1 Node.js veebiraamistike jõudlustesti tulemused aastal 2017 [42]

Jooniselt on näha, et käesoleva töö autori poolt väljavalitud serveripoolsete raamistike seast esines kõige paremini Express, mis suutis vastata keskmiselt 1745 päringule sekundis Ubuntu alamsüsteemis ning 2875 päringule Digital Ocean virtuaalsüsteemis.

Expressile järgneb Sails raamistik, mis suutis vastatv 1554 korda Ubuntu süsteemis ning 772 korda Digital Ocean virtuaalsüsteemis.

Kõige halvemini esines Hapi raamistik, mis suutis tagastada 1094 sõnumit Ubuntu süsteemis ning 688 vastust Digital Ocean virtuaalsüsteemis.

Eelpool selgitatu põhjal võib väita, et Express on jõudlust silmas pidades kõige kindlam valik teenuse serveri poole ehitamiseks.

3.3.2 Kommuuni suurus

Kommuuni suuruse võrdlemine serveripoolsete raamistike puhul on tähtis leidmaks kaasmõtlejaid ning abi probleemide lahendamisel. Lisaks näitab kommuuni suurus lahenduse töökindlust.

Võrdluseks kasutab autor raamistiku allatõmbamiste arvu NPM'ist 2017. aasta aprillikuus (vt Tabel 1). See number näitab, kui palju inimesi raamistiku aktiivselt kasutab ning sinna panustavad.

Tabel 1 Node.js veebiraamistike allalaaditavus 2017. aprillis. [18, 21, 26]

Raamistik	**Allatõmbamiste arv**
Express JS	10 925 693
Hapi JS	279 450
Sails JS	55 952

Tabelist on näha, et Expressi on tõmmatud kordades rohkem alla, kui Hapit või Sails'i. Sellest võib järeldada, et kõrgeima tulemuse saavutanud raamistik on piisavalt suure kommuuniga ning omab rohkelt aktiivseid kasutajaid, mis tõestab antud lahenduse kindlust produktsioonis.

3.3.3 Nõudmised

Vastavalt arendaja eelistustele ning projekti iseloomule võivad raamistiku nõuded olla piiravad või abistavad. Käesoleva töö autor usub, et tarkvaraarenduses peavad otsused langetama arendaja ning raamistik tohib olla vaid tööriist, millega vajalik töö ära teha.

Küsimused, milliseid lisamooduleid, tarkvaraarendusmustreid ning mallide genereerimissüsteeme kasutada, ei tohi olla piiratud.

Tuginedes peatükis 1.2 tehtud Javascripti serveripoolsete raamistike tutvustusele, leiab autor, et Express ning Hapi on oma loomuselt vabad raamistikud, seadmata arendajale piiranguid. Kasutaja saab ise valida malli süsteeme, tarkvaramustri, failide hierarhia ning vajalikud lisamoodulid. Sails vastandub väljatood kahele raamistikule oma nõuete poolest, sest eeldab MVC tarkvaramustri jälgimist ning kindlat failide struktuuri. Lisaks on Sails'i sisse ehitatud palju lisamooduleid, mida ei ole vaja kasutada antud projekti raames.

Tuginedes 1.2 peatükis koostatud tutvustusele, väidab autor, et nõudmiste ning valikuvabaduse osas on Hapi ja Express paremad valikud kui Sails, sest ei sea arendajale piire arendustöö osas.

3.3.4 Väljavalitud serveripoolne Javascripti veebisüsteemide raamistik

Toetudes eelnevale Node.JS veebisüsteemi serveripoolsete raamistike võrdlusele, otsustab autor kasutada projekti loomiseks Express.JS'i.

Käesoleva töö autor leiab, et Express on vastavalt jõudlustestile kõige võimekam kolmest raamistikust ning omab palju aktiivseid kasutajaid, kes arendustöös abistada saavad. Lisaks näitab suur kasutajate arv raamistiku töökindlust. Valikut toetab ka kõnealuse raamistiku nõuded – Ekspress annab arendajale vabad käed moodulite ja tarkvaramustrite osas ning ei sea piiranguid failistruktuuri ülesseadmisel.

3.4 Kliendipoolse veebisüsteemi raamistiku valimine

Järgnevates alampunktides võrdleb autor Angular JS, React JS ning Vue JS raamistikke ning valib välja kliendipoolse raamistiku, mida kasutada näidisprojekti ehitamisel. Võrreldakse raamistiku kiirust, kommuuni suurust ning nõudmisi.

3.4.1 Kiirus

Kliendipoolse raamistiku valimisel on kiirus ning jõudlus tähtis, sest lõppkasutajale peab tema tegevus tunduma intuitiivne ja muredevaba. Kompaktsus ning targad lahendused on käesoleva töö autori silmis tähtsad raamistiku tunnusjooned.

Jõudluse mõõtmiseks kasutab autor Stefan Krause'i poolt läbiviidud Javascripti veebiraamistike jõudlustesti, kus testi läbiviija mõõtis raamistike erinevate tegevuste kiirusi. [43]

Võrdluse aluseks võtab käesoleva töö autor raamistikud Angular v2.0.0-rc5, React v15.3.1 ning Vue v2.0.0-beta1. Testis välja toodud arvud näitavad tegevuse aega millisekundites. Valitud tulemusi testist on võimalik näha tabelist (Vt Tabel 2).

Tabel 2 Javascript veebiraamistike kiirustesti valitud tulemused. [43]

Tegevus / raamistik	**Angular** (tegevuse aeg ms)	**React** (tegevuse aeg ms)	**Vue** (tegevuse aeg ms)
1000 rea tegemine tabelis pärast lehe laadimist	198.067 +/- .91	187.288 +/- .94	171.365 +/- .15
1000 rea uuendamine tabelis	178.451 +/- .98	190.162 +/- .20	68.760 +/- .93
Iga kümnenda rea uuendamine tabelis	11.421 +/- .23	16.401 +/- .07	22.170 +/- .74
Tabeli rea selekteerimine	2.390 +/- .24	5.960 +/- .64	13.301 +/- .06
Kahe rea vahetamine omavahel tabelis	50.162 +/- .81	48.250 +/- .53	19.140 +/- .55
Rea kustutamine tabelis	64.111 +/- .88	67.072 +/- .54	44.090 +/- .77
10 000 rea tegemine tabelis	1914.7063 +/- .86	1839.9629 +/- .79	1712.878 +/- .13

Testi tulemustest on näha, et kõige kiiremad tulemused saavutas enamus tegevustes Vue.JS. Angular ja React olid paremad väiksemates, vähenõudlikes tegevustes, kuid rohkem resurssi nõudvate ülesannete puhul suutis Vue selgelt paremini esineda.

Arvestades, et väiksemate ülesannete puhul ei ole resurssikulu väga suur, leiab käesoleva töö autor, et kiirustesti põhjal on parim raamistiku valik Vue.JS, mis on selgelt kiirem oma konkurentidest suuremate ülesannete täitmisel.

3.4.2 Kommuuni suurus

Kommuuni suurus, nagu ka serveripoolse raamistiku valikul, on tähtis abi ning nõu küsimise pärast, mõttekaaslaste leidmiseks. Eriti kehtib kliendirakenduste puhul, kus õigeid lahendusi samale probleemile võib leiduda väga palju.

Võrdluse aluseks kasutab autor raamistiku allatõmbamiste arvu NPM'ist 2017. aasta aprillikuus (vt Tabel 3). See number näitab, kui palju inimesi raamistiku aktiivselt kasutab ning sinna panustavad.

Tabel 3 Kliendipoolsete veebiraamistike allalaaditavus 2017 aprillis. [31, 37, 41]

Raamistik	**Allatõmbamiste arv**
Angular JS	781 453
React JS	4 098 181
Vue JS	492 097

Antud tulemustest on näha, et allalaadimiste arvu poolest on selgelt teistest üle React JS. Angular JS ning Vue JS on võrdsemate arvudega.

Võib öelda, et kolmest võrreldavas raamistikust on suurima populaarsusega React JS, mis näitab, et Facebook'i poolt arendatud lahendus on kasutuskõlblik ning piisavalt suure kommuuniga. Samas on ka teiste raamistike allatõmbamiste arvud piisavalt suured, et neid arvestada tõsiste raamistikena.

3.4.3 Nõudmised

Kliendipoolse raamistiku valimisel lähtuvalt nõudmistest peab käesoleva töö autor tähtsaks kompaktsust ning valikuvabadust teiste moodulite osas. Ideaalne raamistik peab võimalikult vähe tegema eeldusi kolmandate osapoolte poolt arendatud funktsionaalsuse osas ning olema paindlik.

Tuginedes peatükis 1.3 väljatoodud raamistike tutvustustele, saab öelda, et Angular kasutab rohkem sisseehitatud lahendusi ning on seetõttu ka mahukam. Vue JS ja React JS vastanduvad siinkohal Angularile. Vue mainib vastuvõtlikust ning paindlikust ka

ametlikus tutvustuses, millest võib eeldada, et nende jaoks on väga tähtis teiste moodulitega koos töötamine.

React ning Vue rõhuvad ka komponentidepõhisele ehitusprintsiibile, mis käesoleva töö autori silmis on uuenduslik ning vastuvõetav ehitusmuster. See aitab organiseerida projekti ning hoida kasutajaliidese osasid taaskasutatavana.

Autor leiab, et Vue JS poolt pakutud võimalus ühte faili kogu komponendiga seotud kood kirjutada on vastuvõetav ning aitab struktureerida kirjutatud süsteemi arhitektuuri.

Vastavalt kirjeldatule usub bakalaureusetöö kirjutaja, et React ning Vue JS on sobilikud valikud näidisteenuse loomiseks. Mõlemad raamistikud jagavad sarnaseid printsiipe, Vue JS pakub ka ühte faili komponentide kirjutamist. Angular JS ei ole sobilik, sest implementeerib palju sisse-ehitatud mooduleid, mida arendaja ei pruugi kasutada.

3.4.4 Väljavalitud kliendipoolne Javascripti veebisüsteemide raamistik

Toetudes eelnevale veebisüsteemi kliendipoolse raamistike võrdlusele, otsustab autor kasutada projekti loomiseks Vue JS'i.

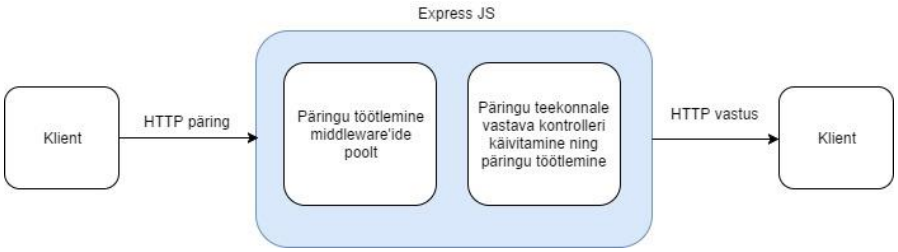
Vue on jõudlustesti põhjal kõige targem valik, sest sooritas palju mahukaid ülesandeid oma konkurentidest kiiremini. Lisaks soosib antud raamistik komponentidepõhist arhitektuuri ning ühte faili koodi kirjutamist. Vue ja React mõlemad ei tee palju eeldusi teiste moodulite osas, samas Angular JS hõlmab palju sisse-ehitatud mooduleid. Kuigi kommuun on kõige suurem React JS'il, leiab töö autor, et ligi 500 000 allatõmbamist ühe kuu jooksul on piisav, et leida kaasmõtlejaid probleemide lahendamisel.

3.5 Tehnoloogiate ning teekide valik

Järgnevates alapeatükkides toob autor välja tähtsamad tehnoloogiad ja teegid, mida kasutati ehitatava halduspaneeli, veebilehekülje ning API loomisel.

3.5.1 Express

Express on käesoleva bakalaureusetöö raames arendatava projekti serveripoolse arenduse tähtsaimaks komponendiks. Antud raamistikku kasutatakse API päringute haldamiseks, autoriseerimise täitmiseks ning kliendipoolse rakenduse serveerimiseks.



Joonis 2 Expressi HTTP päringu teekond läbi *middleware*'ide ja kontrolleri

Tehes päring teenuse rakendusliidesesse, töödeldakse seda Express *middleware*'ide poolt. Pärast seda käivitatakse vastav kontrollerfunktsioon, mis on seotud kindla lõpp-punktiga. Kui kontroller on töö lõpetanud, tagastatakse kliendile vastus. Illustratiivne versioon sellisest funktsionaalsusest on välja toodud joonisel üleval (Vt Joonis 2).

Töö, mida Expressi *middleware*'d teevad, võib olla mitmesugune. Käesoleva projekti raames kasutatakse neid muuhulgas turvalisuse tagamisel. Iga kord, kui klient teeb päringu, vaadatakse päringu päiseid ning sisu, et kindlaks teha *token*'i olemasolu. *Token* on vajalik pääsemaks ligi priviligeeritud informatsioonile. Juhul kui vajalik informatsioon puudub või see ei valideeru, saadetakse kliendile vastav veateade ning päringu töötlemine lõpeb *middleware*'es. Joonisel on näha väljavõtet autoriseerimise *middleware*'st, mis kontrollib asutaja tunnuskoodi ning *token*'i olemasolu (Vt Joonis 3).

```
export default function (req, res, next) {
    const token = req.body.token || req.query.token || req.headers['x-access-token'];
    let userId = req.body.userId || req.query.userId || req.headers['user-id'],
        signInPromise = null;


    if (token) {
        if (token !== configuration.adminKey || !userId) {
            signInPromise = new Promise((resolve, reject) => {
                firebase.auth().signInWithCustomToken(token)
                    .then(response => {
                        userId = response.uid;

                        resolve();
                    })
                    .catch(error => {
                        reject({
                            message: 'Sign in failed',
                            error
                        });
                    });
            });
        } else {
            signInPromise = new Promise(resolve => {
                resolve();
            });
        }
```

Joonis 3 Väljavõte Express *middleware*'st, mis kontrollib *token*'i ning kasutaja tunnuskoodi olemasolu

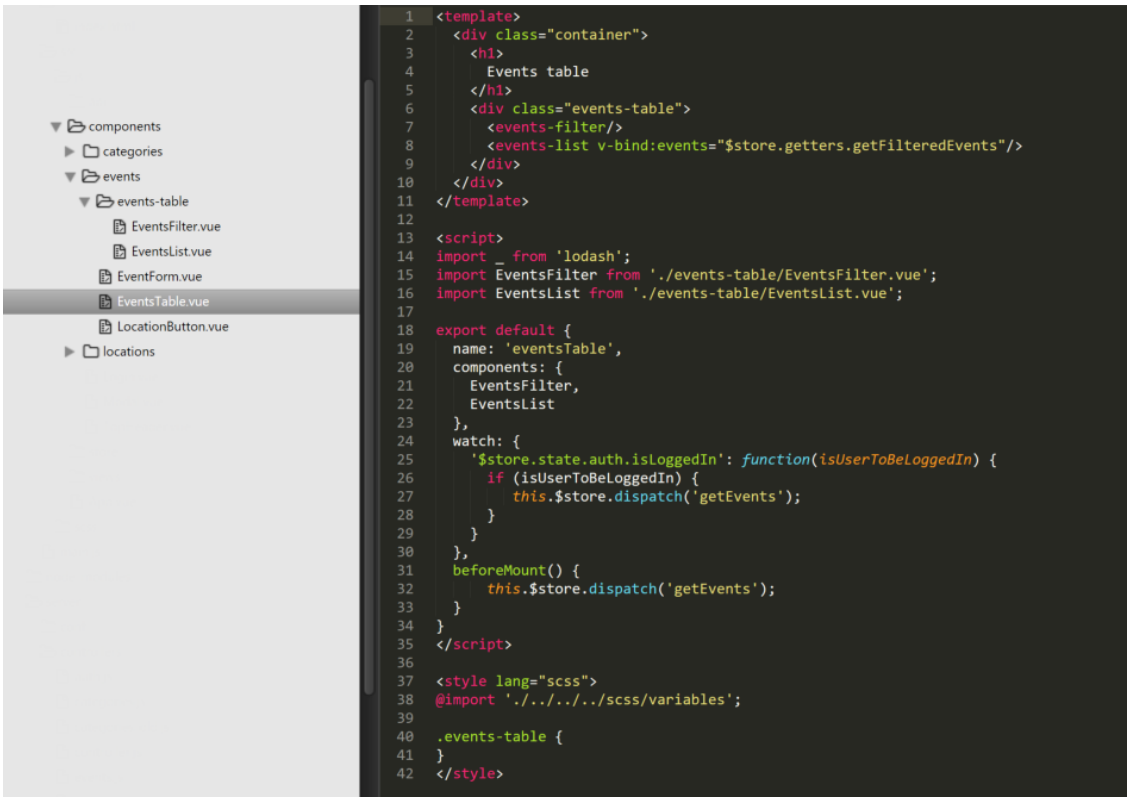
Lisaks rakendusliidesele kasutatakse Expressi ka staatiliste failide serveerimiseks. Sellest tulenevalt on Expressi vaja, et lõppkasutaja näeks kliendipoolset Vue rakendust.

Käesoleva töö autor usub, et Express on võimekas ning sobilik tööriist Javascriptis veebisüsteemide arendamiseks. Antud raamistikul on palju õpetusi ning abivalmis arendajaid, kes töö tegemisel aitavad. Lisaks on kogu süsteemi töötamine arusaadav ning loogiline. *Middleware*- ja kontrollfunktsioonide implementeerimine teeb päringute töötlemise piisavalt võimalusterohkeks, mis antud projektis hädavajalik oli.

3.5.2 Vue JS

Kift projekti kliendipoolsete rakenduste töölepanemiseks kasutatakse Vue JS raamistiku, mis komponentidepõhise ehitusstiili ning kerge informatsiooni manipuleerimise võimekuse tõttu sobib ehitatava projekti arendusse hästi.

Vue JS on implementeeritud nii avalikku veebilehekülge kui ka halduspaneeli. Kõik interaktiivsed toimingud nimetatud süsteemides on toimivad läbi kõnealuse raamistiku. Antud mooduli suureks positiivseks küljeks on ühes failis kogu komponendi koodi hoidmise võimalikkus. Jooniselt on näha, et nii HTML, Javascript kui ka SCSS on ühes failis (Vt Joonis 4). Selline ülevaade toimuvast teeb pikas perspektiivis arendustöö lihtsamaks.



Joonis 4 Vue JS halduspaneeli ürituste tabeli komponent koos alamkomponentidega

Mainitud pildil on võimalik näha ka tüüpilist komponendi struktuuri. Algab fail *template* elemendiga, mille sisse läheb komponendiga seotud HTML. Viimane võib sisaldada ka teisi Vue komponente tähistavaid elemente. Konkreetses näites on välja toodud *events-filter* ja *events-list* komponendid. Seejärel tuleb Javascripti osa, kus antakse komponendile nimi, märgitakse ära teised komponendid ning seadistatakse funktsionaalsus. Näiteks *watch* parameetris jälgitakse *$store.state.auth.isLoggedIn* muutuja väärtust üle rakenduse ning selle muutumisel käivitatakse kaasaantud funktsioon. Faili lõpus on stiilide määramine komponentidele, antud juhul kasutades SCSS süntaksit.

Lisaks struktuurile on Vue JS hea valik ka deklaratiivse renderdamise tõttu. Käesolevas projektis on kliendipoolsetes rakendustes palju muutujaid, mis võivad väärtust vahetada. Deklaratiivse renderdamise tõttu kuvatakse Javascriptis muutunud väärtused koheselt ka DOM'is ning seeläbi näeb ka neid lõppkasutaja. Antud funktsionaalsus tuleb selgelt esile ürituse lisamise kuupäeva valikus halduspaneelis, kus kuupäeva valiku lõppedes muutub tekstikasti sisu, kuid ka selle kohal oleva teksti väärtus. Alloleval joonisel on vasakul pool olev tekst „Start time“ Javascript muutuja, millele pärast tekstikastile sisu

määramist lisatakse lõppu kuupäev ning muudatus on koheselt näha ka kasutajaliideses (Vt Joonis 5).

Joonis 5 Vue deklaratiivse renderdamise esinemine halduspaneels ürituse loomisel

Käesoleva töö autor usub, et Vue oli sobilik valik Javascripti veebirakenduse kasutajaliidese ehitamiseks. Projektis tuli teha otsuseid nagu milliseid teeke kasutada päringute tegemiseks ja komponentide marsruutimiseks, kuid enamasti olid sobivad lahendused Vue koduleheküljelt leitavad. Komponentidepõhise ehituse tõttu oli projekti haldamine kerge ning deklaratiivse renderdamise tõttu tundub kogu dünaamiline informatsioonivahetus rakenduses märkamatu, intuitiivne.

3.5.3 Webpack

Webpack on tööriist, mille abil saab erinevaid Javascripti, SCSS, JSON ning paljude teiste tehnoloogiate mooduleid kokku pakkida üheks staatiliseks failiks. Lisaks on sellega võimalik töödelda eri tüüpi faile enne, kui need koondfaili lisatakse. Kõik see teeb veebiarenduse kiiremaks – Webpack tegeleb failitüüpide kompileerimise, nende vahel seoste loomise ning kokkupakkimisega. [44]

Antud bakalaureusetöö käigus arendatava projekti raames kasutatakse Webpack'i, et töödelda Vue.JS komponentide ja SCSS faile. Lisaks kasutatakse Babel pistikprogrammi muundamaks autori poolt kasutatav Javascripti uuem versioon Ecmascript 6 vanematele veebibrauseritele arusaadavaks Ecmascript 5'ks.

Autor leiab, et Webpack on tähtis tööriist erinevate kompileerimisprotsesside automatiseerimiseks. Võimaldades kasutada arenduses Ecmascript 6't ning SCSS'i, tõstab Webpack autori silmis arenduse kiirust märgatavalt.

3.5.4 Vue Router

Vue Router on Vue.js ametlik teek, mida kasutatakse komponentide vahel liikumiseks vastavalt päritud aadressile. [45]

Käesoleva töö raames oli antud teek kasutusel halduspaneeli ning avaliku veebilehekülje ehitamisel, kuvamaks külastaja päringule vastavaid komponente. Alloleval joonisel on näha, kuidas Vue raamistik ära siduda Vue Router teegiga ning määrata peakomponent *Main* koos ühe alamkomponendiga *CardsList* (Vt Joonis 6).

```
18  Vue.use(VueRouter);
19
20  const routes = [
21      {
22          path: '/',
23          components: {
24              default: Main
25          },
26          children: [
27              {
28                  path: '',
29                  component: CardsList
30              }
31          ]
32      },
```

Joonis 6 Vue Routeri implementeerimine, peakomponendi ning ühe alamkomponendi määramine

Autor leiab, et Vue Router on loodud teenuse raames tähtis tööriist, sest pakub võimalust külastajatel näha informatsiooni, mis just neid huvitab. Olles Vue ametlik marsruutimismoodul, on Vue Router kergesti implementeeritav ning kasutatav.

3.5.5 Vuex

Vuex on teek, mille eesmärgiks on pakkuda oleku haldamist Vue JS rakendustes. Antud tööriist pakub tsentraliseeritud lao objekti, mida kõik Vue komponendid saavad kasutada oleku pärimiseks. See on vajalik selleks, et andmete lugemine ning kirjutamine kliendipoolses rakenduses oleks jälgitav ning ootustele vastav.

Vue rakendusi on võimalik ehitada ka ilma Vuex teegita, kui kasutada atribuute oleku edastamiseks komponentide vahel. Selline lahendus sobib väikeste süsteemide puhul, kus muutuvat infot on vähe. Vuex aga teeb info pärimise ning muutmise lihtsamaks, sest kõik komponendid pääsevad tsentraliseeritud laole ligi, mistõttu ei pea arendaja muretsema atribuutide edastamise pärast. [46]

Autori poolt loodavas teenuses on Vuex kasutusel halduspaneelis ning avalikus veebileheküljes. Näiteks kasutatakse seda halduspaneelis ürituste filtreerimiseks, edastamaks infot aktiivsete filtreerimisparameetrite kohta, läbi mille teised komponendid teavad, milliste ürituste infot antud hetkel kuvada. Alloleval joonisel on näha ürituste alamlao algolek, milleks on Javascripti objekt etteantud atribuutidega. Sellele objektile pääsevad ligi kõik Vue komponendid. (Vt Joonis 7).

```
// initial state
const state = {
    events: {},
    userEvents: {},
    categories: {},
    filter: {
        filterBy: 'title',
        order: 'desc',
        userId: 'all',
        showOnly: {
            field: 'categories',
            value: ''
        }
    },
    eventsUpdated: 0
};
```

Joonis 7 Halduspaneeli ürituste alamlao algolek

Autor leiab, et Vuex on käesoleva töö raames tähtis tööriist. Projekti kliendipoolsed süsteemid sisaldavad suures koguses informatsiooni, mida ilma tsentraalse lao olemasoluta oleks raske hallata. Vuex teeb vajalike andmete hoidmise ning kättesaamise kergemaks, kiirendades oluliselt arenduse kiirust.

3.5.6 Firebase

Firebase on Google poolt arendatav pilve- ja taustateenuse pakkuja. Nende teenuste hulka kuuluvad reaalaja andmebaasid, autoriseerimisteenused, pilvemajutus, analüütika ja palju muud. Teenus annab arendajatele võimaluse kasutada API'sid, läbi mille sooritada erinevaid Firebase'i poolt pakutavaid funktsioone nagu autoriseerimine või andmete kirjutamine andmebaasi. [47, 48]

Käesoleva projekti raames kasutatakse Firebase'i kui autoriseerimisteenust ning andmebaasi. Esimene on fundamentaalne tööriist privaatse API korrektsel töötamisel. Iga päring, mis rakendusliidesesse saadetakse, peab sisaldama *token*'it, mis on saadud

Firebase'ist pärast sisselogimist. Iga privaatpäringu puhul kontrollitakse Firebase'i poolt *token*'i tõesust ning vastavalt resultaadile jätkatakse päringu täitmist või tagastatakse veateade. Lisaks on kasutusel Firebase'i andmebaasi teenus. Kõik halduspaneelis tehtavad päringud kasutavad ära loodud API't, mis vastavalt loeb ja kirjutab andmeid Firebase'i andmebaasi. Antud andmebaasis on info JSON formaadis ning on sealt läbi rakendusliidese alati kättesaadavad. Alloleval joonisel on välja toodud kaks meetodit – kasutaja *token*'i autoriseerimise kontroll ning kasutaja *token*'i kättesaamine vastavalt tema tunnuskoodile (vt Joonis 8).

```
authenticateByToken(token) {
    return this.firebase.auth().signInWithCustomToken(token).then(response => {
        return response.uid;
    })
    .then(userId => {
        return this.getUserObject(userId);
    })
    .then(userObject => {
        return {
            userObject
        };
    })
    .catch(error => {
        return Promise.reject(error.code);
    });
}

getToken(userId) {
    return this.auth.createCustomToken(userId)
        .then(customToken => {
            return customToken;
        })
        .catch(error => {
            return Promise.reject(error.code);
        });
}
```

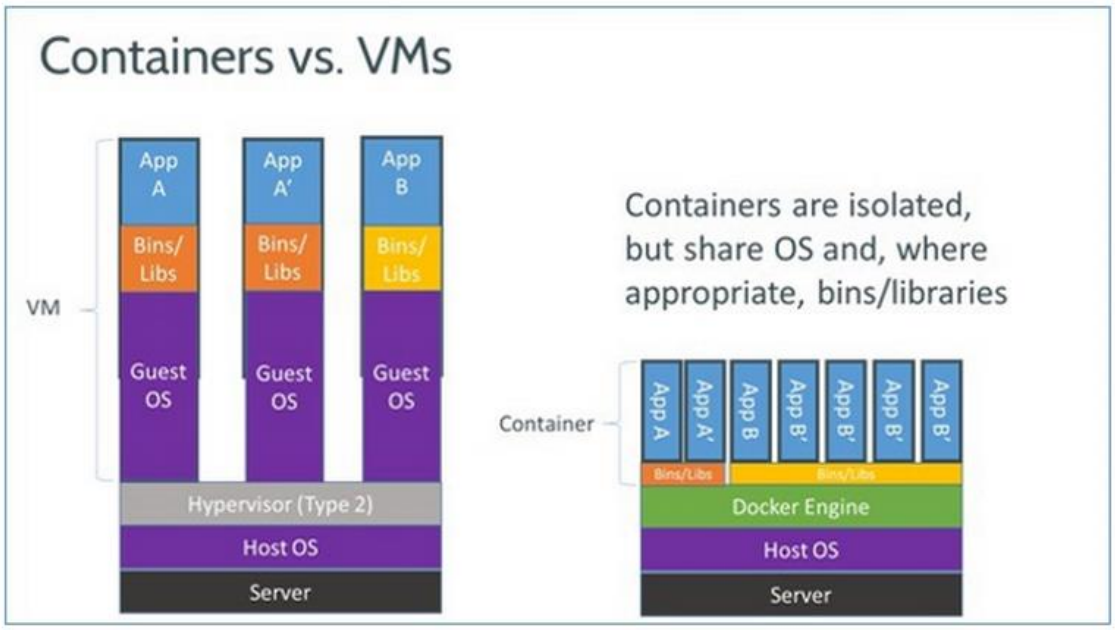
Joonis 8 Firebase'i tokeni kontroll ning kättesaamine vastavalt kasutaja tunnuskoodile

Autor usub, et Firebase on sobiv lahendus reaalajas töötavate lahenduste arendamiseks, kuid antud projektis muutus lisapäringute tegemine tülikaks ning tõi loodavate teenuste kiirust alla. Alternatiivina võib tulevikus kasutusele võtta lokaalse andmebaasi ning eraldi välja töötatud autoriseerimissüsteemi.

3.5.7 Docker

Docker on tarkvara pakendamiseks loodud süsteem, mis lubab ühes serveris jooksutada samaaegselt erineva tarkvaraga alamsüsteeme ehk konteinereid. Docker on populaarne arendajate ja süsteemiarhitektide seas, sest laseb automatiseerida tarkvara ülesseadmist erinevatel operatsioonisüsteemidel. [49] Näiteks saab selle tööriistasid kasutades jooksutada täpselt samades tingimustes töötavaid rakendusi Windows'i, Mac OS kui ka

Linux operatsioonisüsteemidel. Alloleval pildil on toodud võrdlus traditsiooniliselt sarnase eesmärgi saavutamiseks mõeldud virtuaalmasinate ning Dockeri vahel (Vt Joonis 9). Pildilt on näha, et virtuaalmasinad tekitavad süsteemis duplikaat operatsioonisüsteemid, kuid Docker seda ei tee, hoides kokku ruumi ning süsteemi riistvaralisi resursse.



Joonis 9 Dockeri võrdlus virtuaalmasinatega [49]

Lisaks mainitule pakub Docker ka teenust nimega Docker Cloud, läbi mille saab arendaja ehitada täisautomaatse süsteemi tarkvara ülesseadmiseks soovitud keskkonnas. Docker Cloud võtab vastu Docker konteineri konfiguratsioonifaili ning vastavalt seadistustele viib läbi erinevaid etteantud toiminguid. Näiteks saab arendaja automaatselt suletud keskkonnas tarkvara testida ning üles laadida enda serverisse, kus see tööle pannakse. [50, 51]

Käesolevas projektis on Docker kasutusel tarkvara pakendamisel ning automaatsüsteemi loomisel. API ja halduspaneel positsioneeruvad ühes konteineris, millele on installeeritud Node 6. Konfiguratsioonis on veel määratud, et Dockeri käivitamisel laetaks alla kõik vajalikud Javascript'i moodulid ning tegevuse lõppedes aktiveeritaks teenus. Sama protsess toimub ka avaliku veebilehe konteineris, kuid Javascript'i moodulid, mida installeeritakse, on teised. Loetletud tegevuste tagajärjel hakkab operatsioonisüsteemis jooksma kolm teenust- API, halduspaneel ning avalik veebilehekülg.

Arendustöö automatiseerimiseks on kasutusel Docker Cloud. Pärast igat koodi üleslaadimist projekti Git repositooriumi *master* harusse, loeb Docker Cloud projekti konfiguratsioonifaile ning ehitab valmis failisüsteemid, mille abil tekitada produktsiooni serveris vajalikud konteinerid. Tulemuseks on täisautomaatne üleslaadimisega süsteem.

Käesoleva töö autori hinnangul on Docker hea vahend tarkvaraliste tegevuste automatiseerimiseks. Võimalus installeerida automaatselt vajaminevad programmid, need üles laadida ja käima panna lihtsustab arendaja tööd ning hoiab kokku palju aega.

3.5.8 Redis

Redis on vabavaraline mälusisene andmete salvestamiseks mõeldud tööriist. Seda kasutatakse vahemäluna, andmebaasina ning sõnumite edastajana. [52]

Käesolevas projektis on Redis kasutusel API's andmete salvestamiseks vahemällu, et neid rakendusliidese avaliku lõpp-punkti külastamisel kiiremini pärijatele saata. Iga 5 minuti tagant teeb rakendusliidese server päringu Firebase'i, kust saadakse kogu vajalik informatsioon. Vastus salvestatakse lokaalsesse Redis serverisse. Külastades avalikku lõpp-punkti ürituste, kategooriate ning asukohtade saamiseks, loeb API server vastused Redise serverist, mistõttu jääb ära teekond Firebase'i ning hoitakse kokku resurssi ja aega. Alloleval joonisel on näha väljavõtet Redise teenuse implementeerimisest (Vt Joonis 10).

```
const redisClient = redis.getClient();

/*
    Return events for public usage by getting them from Redis
    or Firebase
*/
router.get('/kift', (req, res) => {
    redisClient.getAsync('kift-content')
        .then(redisResponse => {
            return new Promise((resolve, reject) => {
                let content = JSON.parse(redisResponse);

                content.canBeFetched ? resolve(content) : reject('no content');
            });
        })
        .then(contentObject => {
            res.send({
                success: true,
                response: contentObject
            });
        })
        .catch(() => {
            getEvents(req, res)
                .then(contentObject => {
                    res.send({
                        success: true,
                        response: contentObject
                    });
                })
                .catch(() => {
                    res.send({
                        success: false,
                        error: 'Could not get events'
                    });
                });
        });
});
```

Joonis 10 Redise kasutamine ürituse pärimisel API avalikus lõpp-punktis

Jooniselt on näha, et külastades avalikku API lõpp-punkti, proovib server saada asünkroonselt Redis serverist *kift-content* väärtuse, mis sisaldab endas kogu avalikkusele mõeldud informatsiooni. Juhul kui päring on edukas, muundatakse vastus Javascript objektiks ning kontrollitakse *canBeFetched* väärtust. Viimase tõesuse puhul saadetakse vastus kliendile. Kui kirjeldatud protsessis peaks tekkima viga, päritakse üritused *getEvents* funktsiooni abil Firebase'st ning tagastatakse kliendile. Vea esinemise korral tagastatakse veateade.

Käesoleva töö autor leiab, et Redis on antud teenuse raames sobiv tööriist päringutele vastamise kiiruse tõstmiseks ja serveri koormuse langetamiseks, sest läbi mälusisese salvestamise ei edastata kliendi päringuid Firebase'i ning ei kutsuta välja erinevaid funktsioone andmete töötlemiseks.

3.5.9 Facebook Graph API

Facebook Graph API on rakendusliides, läbi mille pääseb ligi Facebook'is olevale informatsioonile. Vastavalt õigustele on võimalik lugeda ning kirjutada informatsiooni postituste, reklaamide, ürituste, gruppide ning piltide kohta. See on peamine tööriist arendajatele Facebook'i teenuste implementeerimiseks oma rakendusse. [53]

Käesolevas projektis on Facebook Graph API kasutamise lihtsustamiseks rakendatud Fbgraph Node.js teeki. [54] Antud moodulit kasutatakse Facebook'ist kindlate ürituste kohta käiva info allalaadimiseks. Kõnealusesse sotsiaalvõrgustikku saadetakse informatsioon kindlate asukohtade tunnusnumbritega ning vastuseks saadakse antud asukohaga seotud tulevased üritused. Need üritused töödeldakse Node.js serveris Kift teenusele sobivaks ning salvestatakse andmebaasi juhul, kui need juba ei eksisteeri. Alloleval pildil on näha väljavõtet Graph API kasutamisest (Vt Joonis 11). Andes ette päringus meid huvitavad atribuudid, tagastatakse detailne informatsioon ürituse kohta (Vt Joonis 12).

```
getPicture: function (graph, returnArray, fetchedPageNr, fetchedEventNr, access, callback){
  //GET THE event ID OF EACH EVENT
  var eventID = returnArray[fetchedPageNr].data[fetchedEventNr].id;
  //GET THE LINK FOR THE PICTURE
  graph.get(eventID  +   "/?fields=cover,attending_count,interested_count",{access_token: access}, function(err, res) {
    //ADD THE PICTURE LINK TO THE RETURN ARRAY
    returnArray[fetchedPageNr].data[fetchedEventNr].picture =  res.cover.source;
    returnArray[fetchedPageNr].data[fetchedEventNr].attending_count =  res.attending_count;
    returnArray[fetchedPageNr].data[fetchedEventNr].interested_count =  res.interested_count;
    //CHECK IF THE PICTURES HAVE BEEN FETCHED FOR ALL THE EVENTS:
     if(fetchedPageNr == returnArray.length-1 && fetchedEventNr == returnArray[fetchedPageNr].data.length-1){
      //RETURN THE EVENTS LIST WITH PICTURES
      callback(returnArray);
    }
  });
},
```

Joonis 11 Väljavõte Facebook Graph API rakendamisest soovitud ürituste pärimiseks


```
{
    description: Let's make some noise!,
    end_time: "2017-11-09T23:30:00+0200",
    name: "Dirkschneider at Rock Cafe",
  - place: {
        name: "Rock Cafe",
      - location: {
            city: "Tallinn",
            country: "Estonia",
            latitude: 59.425816451089,
            longitude: 24.779609441757,
            street: "Tartu mnt 80d",
            zip: "10112"
        },
        id: "629554980539341",
        attending_count: 28,
        interested_count: 125
    },
    start_time: "2017-11-09T19:00:00+0200",
    id: "314805678936969",
    picture: "https://scontent.xx.fbcdn.net/v/t1.0-9/17884024_771219879709456_7625252365839947330_n.jpg?oh=33a911748c34f9a2f52254d3b7d55f35&oe=597CCEDD",
    attending_count: 28,
    interested_count: 125
},
```


Joonis 12 Facebook Graph API päringu vastus ühe ürituse kohta

Autori silmis on Facebook Graph API antud projekti raames sobiv lahendus ürituste kättesaamiseks Facebook'ist, sest pakub piisavas koguses funktsionaalsust, läbi mille teenuse jaoks vajalikke üritusi alla laadida. Autor leiab, et ilma Facebook Grap API'ta, ei oleks loodud teenus piisavalt automaatne, tehes kõnealuse mooduli hädavajalikuks.

4. Valminud teenus

Antud bakalaureusetöö käigus valminud projekti eesmärk oli luua terviklik veebilahendus ning API, mis koondab ühtsesse süsteemi erinevad Tallinnas toimuvad meelelahutusüritused vastavalt teenuse omanike äranägemisele. Valminud lahendus on automaatne, tõmmates Facebook'ist alla kindlaks määratud meelelahutusasutuste üritusi. Süsteem määrab üritustele kategooriad, mille alusel neid tagastatakse rakendusliideses ning kuvatakse avalikus veebis. Lisaks on halduspaneelis võimalik tekitada uusi üritusi, mis ei ole seotud Facebook'iga. Valminud avalik veebilehekülg on kasutamiseks kättesaadav aadressil www.kift.place, halduspaneel on külastatav aadressil www.kift.place:1337. Koostöös projekti arendajaga on kolmandatel osapooltel võimalik kasutada valminud rakendusliidest enda teenustes. [55, 56]

Valminud projekti lähtekood ei ole mõeldud avaldamiseks.

Järgnevates alapeatükkides esitleb autor valminud teenust läbi illustratiivse materjali, kontrollib nõuetele vastavust, toob välja kasutatud vahendid ning pakub välja arenduskohti.

4.1 Kasutatud vahendid

Autor kasutas projekti realiseerimiseks järgnevaid vahendeid:

- Sublime Text 3 tekstitöötlusprogramm – tasuline tarkvara, mis lihtsustab koodi kirjutamist läbi abistavate pistikprogrammide ning klaviatuuri lühiteede pakkumise. [57]
- Git Bash – käsurea tööriist, läbi mille projekti lähtekoodi üles laadida repositooriumisse, hallata kolmandate osapoolte teeke ning käivitada arenduse käigus tarkvara. [58]

Loodava veebisüsteemi katsetamiseks ja testimiseks kasutati järgnevaid vahendeid:

- Google Chrome versioon 57.0.2987.133 [59]

- Postman v4.10.7 [60]
- LG Nexus 5, Android 7.1.1

Teenuse loomisel kasutati järgnevaid programmeerimiskeeli:

- Javascript ES6, ES5
- SCSS
- HTML 5
- YAML
- Bash

4.2 Rakendusliides

Valminud teenuse üks alamosasid on rakendusliides, mida kasutavad teenuse halduspaneel ning avalik veebilehekülg. Lisaks saavad API’t kasutada kolmandad osapooled, kui neil on selleks vajalikud load ning õigused.

Rakendusliidese privaatsed lõpp-punktid ei kuulu turvalisuse huvides avalikustamisele.

Rakendusliidese ainuke avalik lõpp-punkt asub aadressil http://www.kift.place:1337/api/public/kift. [61] Vastus koosneb kõikidest avalikest kategooriatest, asukohtadest ning üritustest JSON formaadis.

4.3 Halduspaneel

Halduspaneel on loodud eesmärgiga pakkuda teenuse omanikele ning potentsiaalsetele koostööpartneritele ühtset süsteemi ürituste haldamiseks. Süsteemis saab lisada, muuta ja kustutada üritusi, kategooriaid ning asukohti. Järgnevalt tutvustab autor halduspaneeli põhivaateid.

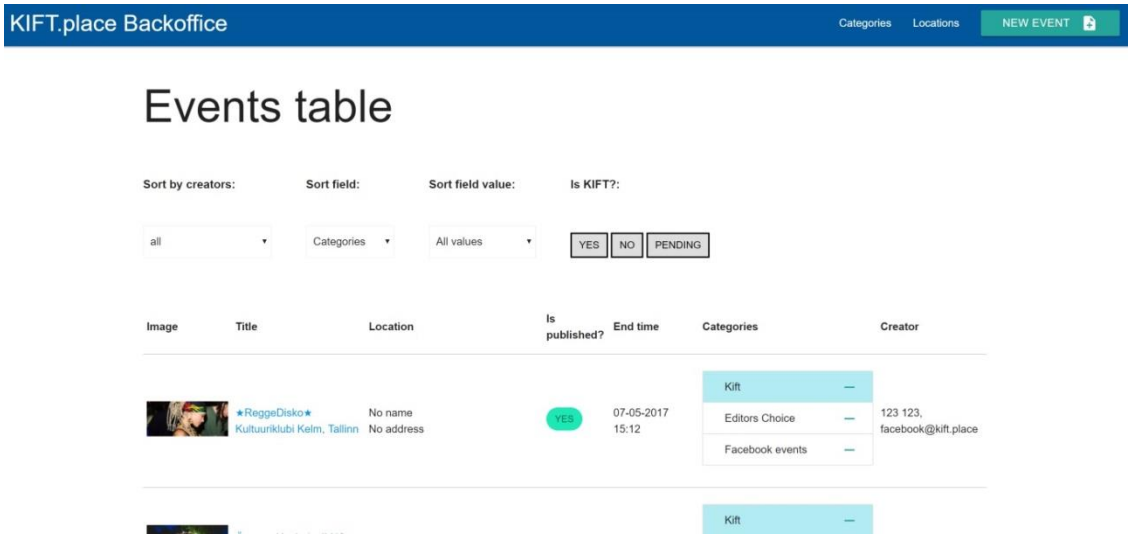
4.3.1 Halduspaneeli sisenemine

Halduspaneeli kasutamiseks on vajalik teenuse omaniku poolt manuaalselt kasutaja tegemist. Omades kasutajat, on võimalik süsteemi õigete andmetega siseneda (Vt Joonis 13).

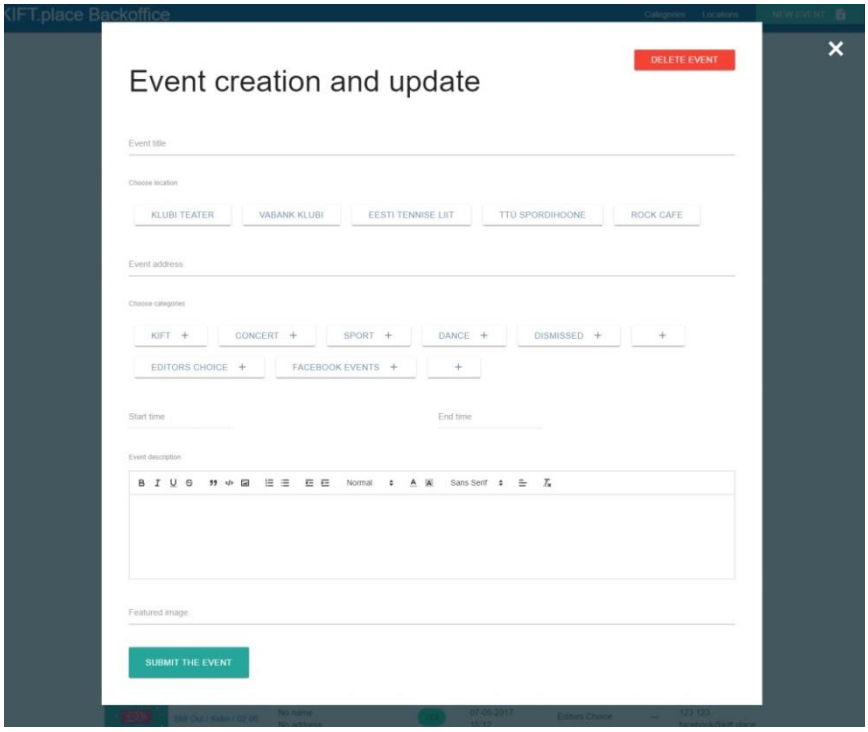
Joonis 13 Halduspaneeli sisenemise vorm

4.3.2 Ürituste haldamine

Ürituste haldamiseks on süsteemis kaks vaadet – nimekirja kujul kõikide ürituste vaade, mida saab filtreerida vastavalt looja, kategooria ning asukoha järgi ning ürituse lisamise ja muutmise vaade, kus saab redigeerida kogu objektiga seotud informatsiooni (Vt Joonis 14, 15).



Joonis 14 Ürituste üldvaade

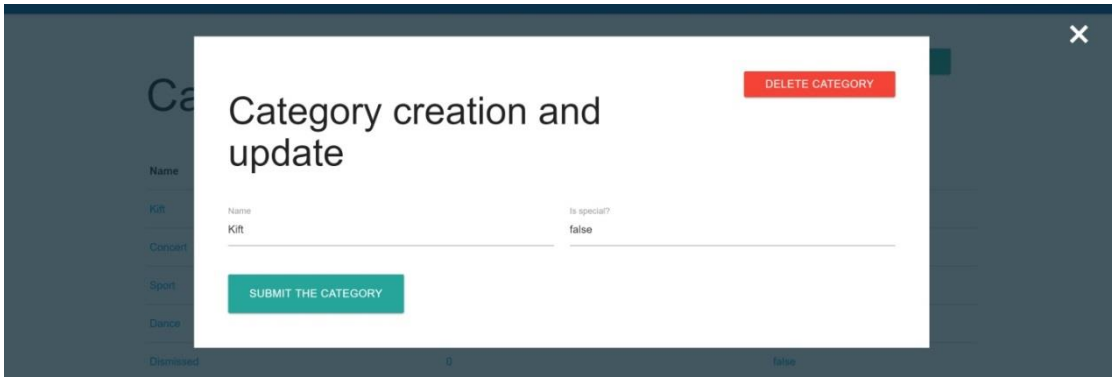


Joonis 15 Ürituste loomise ning muutmise vaade

4.3.3 Kategooriate haldamine

Kategooriate haldamiseks on halduspaneelis vaade, kus saab näha nimekirja olemasolevatest kategooriatest ning lisada uusi kategooriaid (Vt Joonis 16, 17).

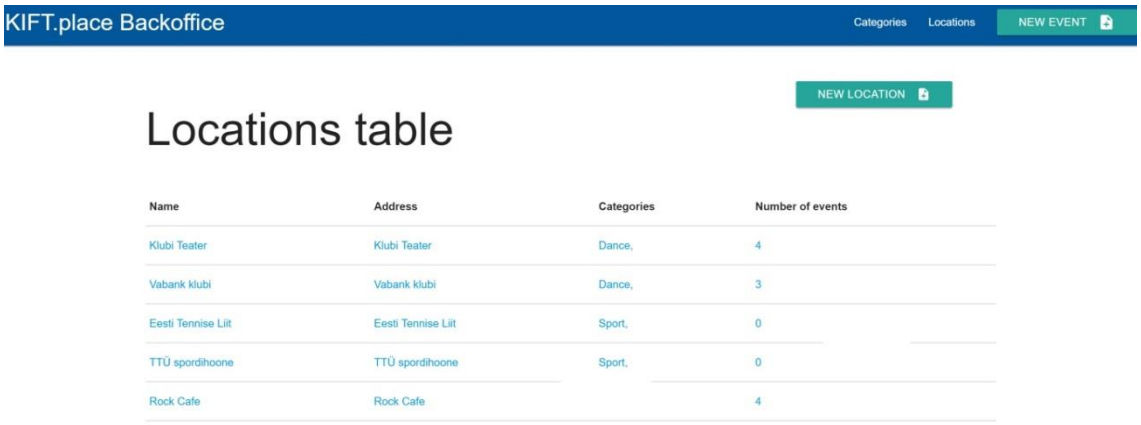
Joonis 16 Kategooriate üldvaade - nimekiri kategooriatest



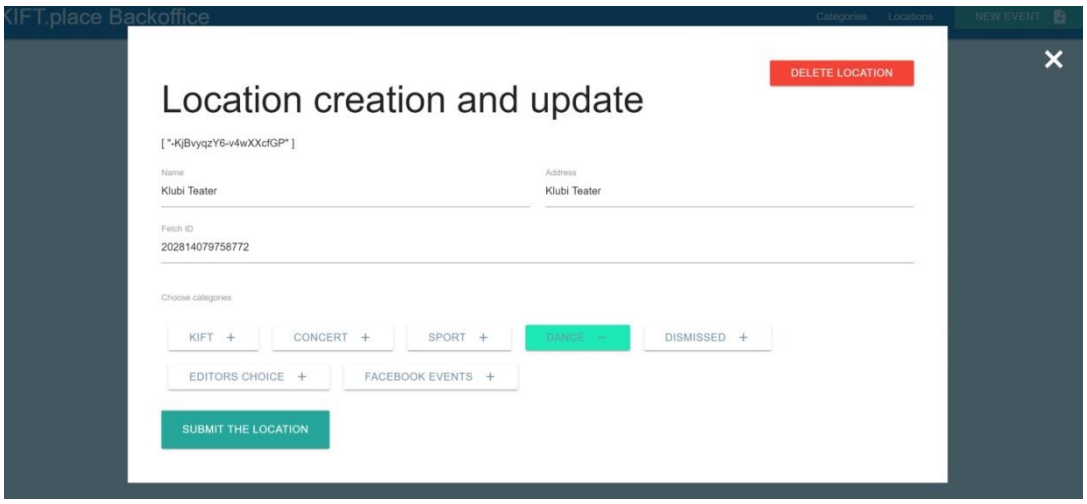
Joonis 17 Kategooria lisamise ning muutmise vaade

4.3.4 Asukohtade haldamine

Asukohtade haldamiseks on sarnaselt kategooriatele nimekirja ning lisamise- ja muutmise vaade (Vt Joonis 18, 19). Asukohta lisades saab valida ka kategooriad, millega antud asukoht seotud on.



Joonis 18 Asukohtade üldvaade



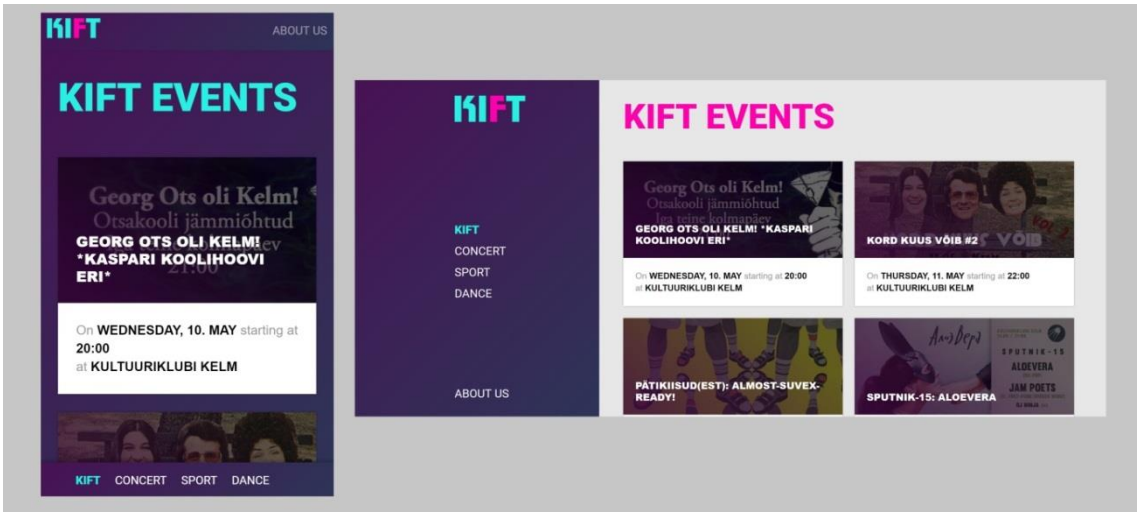
Joonis 19 Asukoha lisamise ja muutmise vaade

4.4 Avalik veebilehekülg

Loodud teenuse avalik veebilehekülg on mõeldud kasutuseks kõikidele inimestele, kes soovivad kerge vaevaga leida Tallinnas üritusi, seda nii mobiilis kui ka arvutis. Järgnevalt on välja toodud veebilehekülje põhivaated.

4.4.1 Üld- ja kategooria vaade

Kõnealune vaade on Kift.place lehekülje peamine vaade. Sellelt on näha vastava kategooria üritused ning sellega seotud informatsioon (Vt Joonis 20). Üritusele klikkides avaneb konkreetse ürituse moodulaken.



Joonis 20 Avaliku veebilehekülje üld- ja kategooria vaated mobiilis ning arvutis

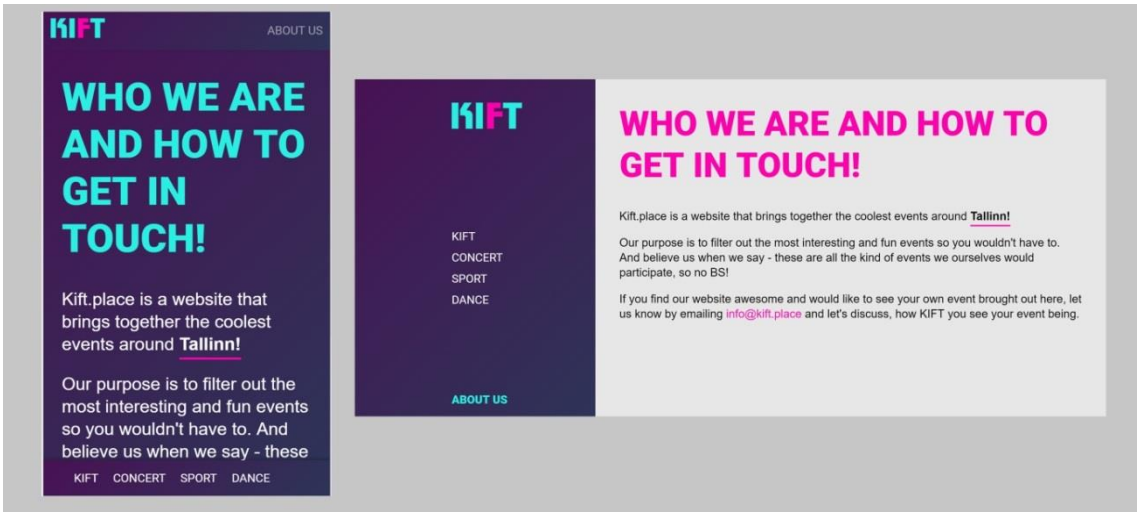
4.4.2 Ürituse detailvaade

Ürituse detailvaateks on moodulaken, mis koosneb ürituse pealkirjast, asukohast, toimumisajast, tunnuspildist ning detailkirjeldusest. Alltoodud illustratsioonil on näha kõnealuse vaate mobiilvaadet ning arvuti versiooni (Vt Joonis 21).

Joonis 21 Ürituse detailvaade mobiilis ning arvutis

4.4.3 Sisuvaade

Käesoleva bakalaureusetöö raames loodud teenuse avaliku veebilehekülje sisuvaade on kasutusel „Meist“ osas, kus kirjutatakse veebilehekülje eesmärgist ning edastatakse huvilistele kontaktandmed. Joonisel on näha vaadet mobiilis ning arvutis (Vt Joonis 22).



Joonis 22 Teenuse avaliku veebilehekülje sisuvaade mobiilis ja arvutis

4.5 Loodud teenuse vastavus esitatud nõuetele

Autor leiab, et kõik projektile algselt esitatud nõuded said täidetud. Järgnevates alapeatükkides on nõuete täituvus toodud välja detailsemalt.

4.5.1 Halduspaneeli nõuded

Autori poolt loodud halduspaneeli nõuetest said kõik täidetud. Tuleb märkida, et meeldetuletuse e-posti saatmine admin kasutajatele töötab käesoleva töö kirjutamise ajal vaid testkeskkonnas, tulenevalt serveri piirangutest. Detailsemat ülevaadet nõuete täitmisest on näha allolevast tabelist (Vt Tabel 4).

Tabel 4 Halduspaneeli nõuded ning nende täitmise lõppstaatus

Teenuse kasutamiseks on vaja kasutajakontot	Täidetud. Halduspaneeli saab siseneda vaid läbi sisenemisvormi, mis nõuab korrektsed e-posti aadressi ning parooli.
Teenuse kasutajaid saavad lisada vaid teenuse loojad	Täidetud. Kasutajakonto saavad luua vaid projektiga seotud inimesed manuaalselt andmebaasis.
Teenusel on kahte tüüpi kasutajad	Täidetud. Eksisteerib tavakasutaja ning admin.
Teenuse kasutamiseks peab kasutaja sisestama enda e-posti ning parooli	Täidetud
Teenus peab valede andmete sisestamisel kasutajat sellest teavitama	Täidetud. Sisenemisvorm kuvab punaselt veateate.
Tavakasutaja peab saama näha enda üritusi ning neid redigeerida, kustutada	Täidetud.
Admin kasutaja peab saama näha kõikide kasutaja üritusi ning neid redigeerida, kustutada.	Täidetud. Admin kasutajal on võimalik rippmenüüst valida kasutaja, kelle üritusi soovitakse näha.
Admin kasutaja peab saama näha kõiki süsteemis olevaid kategooriaid ning neid redigeerida, kustutada	Täidetud. Admin kasutajal on võimalik minna kategooriate lehele.
Admin kasutaja peab saama näha kõiki süsteemis olevaid asukohtasid ning neid redigeerida, kustutada.	Täidetud. Admin kasutajal on võimalik minna asukohtade lehele.
Admin kasutaja peab saama lisada uusi üritusi, kategooriaid ning asukohtasid.	Täidetud
Asukohtade, ürituste ning kategooriate loomisel ning redigeerimisel peab olema võimalik muuta kogu objekti kohta käivat informatsiooni	Täidetud. Kogu vajalikku informatsiooni nagu pealkiri ja kirjeldus, on autoril võimalik muuta.
Üritusi peab saama filtreerida kategooria, asukoha ning avalikustamise järgi.	Täidetud. Kasutaja peab valima ühest rippmenüüst filtreerimisaluse ning teisest filtreerimisaluse väärtuse.
Halduspaneel peab automaatselt iga 30 minuti tagant uuendama Facebookist andmeid	Täidetud. Andmeid uuendatakse iga 30 minuti tagant. Facebookist tulnud üritus, mida admin kasutaja on eraldi redigeerinud, ei kuulu uuendamisele.
Halduspaneel peab automaatselt saatma igale admin kasutajale kord nädalas e-kirja vasutusteatega.	Täidetud testkeskkonnas. Iga nädala uus vastutav saab e-kirja, kus teavitatakse teda tuleva nädala kohustustest.

4.5.2 Avaliku veebilehekülje nõuded

Autor leiab, et kõik avaliku veebileheküljega seotud nõuded said täidetud. Nõuete detailsemat täidetavust on näha allolevas tabelis (Vt Tabel 5).

Tabel 5 Avaliku veebilehekülje nõuded ning nende lõppstaatus

Teenus peab olema kättesaadav aadressil www.kift.place	Täidetud. Veebilehekülg on kättesaadav.
Teenuse informatsioon peab olema vastavuses halduspaneeli informatsiooniga	Täidetud. Muutes informatsiooni halduspaneelis, muutub see ka 5 minuti jooksul veebileheküljel.
Teenuse leheküljel peab nähtaval olema kategooriate menüü	Täidetud
Teenuse leheküljel peab nähtaval olema sektsioon loojate tutvustusega	Täidetud
Teenuse lehekülje kategooriate lehekülg peab sisaldama antud kategooriaga seotud üritusi	Täidetud
Ürituse vaates peab selgelt olema välja toodud toimumiskoht- ja aeg, pealkiri, kirjeldus, tunnuspilt olemasolul	Täidetud
Teenuse lehekülg peab olema responsiivne-kasutatav arvutis ja mobiilis.	Täidetud

4.5.3 Rakendusliidese teenuse nõuded

Autor leiab, et rakendusliidesele seatud eesmärgid said täidetud. Nende täitumist on detailsemalt näha allolevas tabelis (Vt Tabel 6).

Tabel 6 Rakendusliidese teenuse nõuded ning nende lõppstaatus

Teenus omab avalikku ja privaatset poolt	Täidetud. Teenusel on üks avalik lõpp-punkt ning erinevad privaatsed lõpp-punktid halduspaneeli jaoks.
Avalik pool omab lõpp-punkti, mis tagastab avalikuks näitamiseks sobivad andmed.	Täidetud. Avalik lõpp-punkt saadab pärijale andmed asukohtade, kategooriate ning ürituste kohta.
Privaatne pool on kasutatav vaid sisseloginud kasutaja tokeni põhjal või salajase admin võtme abil.	Täidetud
Ebakorrektse või puuduva tokeni puhul tagastatakse veateade	Täidetud
Privaatne pool peab pakkuma ürituste, kategooriate ja asukohtade lisamist, muutmist ja kustutamist.	Täidetud

4.6 Ettepanekud edasiseks arenduseks

Kõnealuse teenuse autor usub, et püstitatud eesmärgid tarkvara loomisel said täidetud, kuid on kohti, kus saaks asju paremini teha.

- Andmebaasisüsteemi väljavahetamine – kasutatud lahendus, Firebase, on andmete salvestamiseks hea lahendus rakendustele, mis nõuavad reaalajas töötamist. Käesoleva projekti juhul sellist vajadust ei ole ning nimetatud süsteemi väljavahetamisel lokaalse andmebaasi vastu võidaks päringu tegemiselt ajaliselt juurde.
- Vue JS serveripoolne renderdamine – projekti kirjutamise käigus leidis töö autor võimaluse Vue rakendus valmis renderdada serveri poolt, mis saadetakse kliendi poolele. Selline funktsionaalsus annab juurde otsingumootorites positsioneerides ning on taolise teenuse puhul hädavajalik. [62]
- Üritustele, mis nõuavad piletiraha, tuleks lisada märge. Sellise süsteemi väljaehitamine nõuab tekstide filtreerimist Facebook’ist, kuid on tervitatav, sest edastab külastajatele täpsemat informatsiooni.

5. Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli tutvuda erinevate kliendi- ja serveripoolsete Javascripti veebisüsteemide ehtamiseks mõeldud raamistikega, neid võrrelda ning sobivaimate alusel luua avalikuks kasutamiseks mõeldud ürituste portaal Kift.place koos kolmandate osapoolte poolt kasutatava rakendusliidese ning halduspaneeliga.

Töö käigus tutvuti Express JS, Hapi JS ning Sails JS serveripoolse Javascripti raamistikega, mis on mõeldud veebisüsteemide ehitamiseks. Lisaks vaadeldi sama programmeerimiskeele kliendipoolseid raamistikke – Angular JS, React JS ning Vue JS. Kogutud informatsiooni põhjal sooritati võrdlus, mille põhjal valiti välja autori silmis sobivaimad lahendused teenuse loomiseks.

Autori eesmärk leida erinevusi vaadeldud raamistikes ning sobivaimate põhjal realiseerida ürituste portaal koos rakendusliidese ning halduspaneeliga sai täidetud. Valitud serveripoolne raamistik, Express JS, oli võrdluse põhjal kõige võimekam ning omas suurimat kasutajate arvu. Kliendipoolne raamistiku valik, Vue JS, oli autori silmis sobiv komponentidepõhise ehitusstiili, jõudluse ning valikuvabaduse pärast teiste moodulite osas. Lisaks võrreldud raamistikele tutvustati töö käigus teisi erinevaid teenuse loomisel tähtsat rolli kandnud tehnoloogiaid nagu Docker ja Redis.

Valminud teenus koondab enda alla kolm alamteenust- avalikuks kasutamiseks mõeldud veebilehekülg, süsteemi administreerimiseks mõeldud halduspaneel ning rakendusliides, mille abil kolmandad osapooled saavad kokkuleppe korral kasutada Kift.place teenuseid. Avalik veebilehekülg on kättesaadav aadressil www.kift.place. [55]

Kasutatud kirjandus

[1] Rakendusliides – Vikipeedia [WWW] https://et.wikipedia.org/wiki/Rakendusliides (11.05.2017)

[2] What Are Access Tokens? – Microsoft Technet [WWW] https://technet.microsoft.com/en-us/library/cc759267(v=ws.10).aspx (11.05.2017)

[3] Data Binding – Wikipedia [WWW] https://en.wikipedia.org/wiki/Data_binding (11.05.2017)

[4] Berg, L. Middleware: THE core of node.js backend apps [WWW] https://hackernoon.com/middleware-the-core-of-node-js-apps-ab01fee39200 (11.05.2017)

[5] Angular JS, Google – Google Plus [WWW] https://plus.google.com/+AngularJS/posts/aZNVhj355G2 (11.05.2017)

[6] Netkachov, A. Model-View-Controller(MVC) in Javascript [WWW] https://alexatnet.com/model-view-controller-mvc-in-javascript (11.05.2017)

[7] Callback (computer programming) – Wikipedia [WWW] https://en.wikipedia.org/wiki/Callback_(computer_programming) (11.05.2017)

[8] Object-relational mapping – Wikipedia [WWW] https://en.wikipedia.org/wiki/Object-relational_mapping (11.05.2017)

[9] What is Document Object Model? – W3 [WWW] https://www.w3.org/TR/DOM-Level-2-Core/introduction.html (11.05.2017)

[10] Jhangra, N. The Rise and Rise of Javascript [WWW] https://www.linkedin.com/pulse/rise-javascript-nitij-jhangra (08.05.2017)

[11] The state of the Octoverse 2016 - GitHub Octoverse [WWW] https://octoverse.github.com (08.05.2017)

[12] Brehm, S. Isomorphic Javascript: The Future of Web Apps [WWW] https://medium.com/airbnb-engineering/isomorphic-javascript-the-future-of-web-apps-10882b7a2ebc (08.05.2017)

[13] NPM Features - NPM [WWW]
https://www.npmjs.com/features (08.05.2017)

[14] Mardan, A. Express.js FUNdamentals: An Essential Overview of Express.js [WWW]
https://webapplog.com/express-js-fundamentals (08.05.2017)

[15] Morgan, A. The Modern Application Stack – Part 3: Building a REST API Using Express.js [WWW] https://www.mongodb.com/blog/post/the-modern-application-stack-part-3-building-a-rest-api-using-expressjs (08.05.2017)

[16] Yang, C. Express, Koa, Meteor, Sails.js: Four Frameworks Of The Apocalypse [WWW]
https://www.toptal.com/nodejs/nodejs-frameworks-comparison (08.05.2017)

[17] Most depended-upon packages – NPM [WWW]
https://www.npmjs.com/browse/depended (08.05.2017)

[18] Express – NPM [WWW] https://www.npmjs.com/package/express (08.05.2017)

[19] Arora, S. Node.js Frameworks: The 10 Best for Web and Apps Development [WWW]
http://noeticforce.com/best-nodejs-frameworks-for-web-and-app-development
(08.05.2017)

[20] Cabot Technology Solution. Creating RESTful API with Node.js: Hapi vs. Express [WWW] https://medium.com/@cabot_solutions/creating-restful-api-with-node-js-hapi-vs-express-ccb97a776c02 (08.05.2017)

[21] Hapi – NPM [WWW] https://www.npmjs.com/package/hapi (08.05.2017)

[22] J, Perinovic. Node.js Framework Comparison: Express.js vs Hapi.js [WWW]
https://objectpartners.com/2016/12/22/node-js-framework-comparison-express-js-vs-hapi-js (08.05.2017)

[23] Node Web frameworks comparison: Express, hapi and Sails - Pluralsight [WWW]
https://www.pluralsight.com/blog/software-development/node-web-frameworks
(08.05.2017)

[24] Sails features - Sails [WWW] http://sailsjs.com/features (08.05.2017)

[25] Socket.io – Github [WWW] https://github.com/socketio/socket.io (08.05.2017)

[26] Sails – NPM [WWW] https://www.npmjs.com/package/sails (08.05.2017)

[27] AngularJS - Superheroic Javascript MVW Framework – AngularJS [WWW] https://angularjs.org (09.05.2017)

[28] Angular 2 Data Binding – Tutorialspoint [WWW] https://www.tutorialspoint.com/angular2/angular2_data_binding.htm (09.05.2017)

[29] NgModules – Angular Guide [WWW] https://angular.io/docs/ts/latest/guide/ngmodule.html (09.05.2017)

[30] Nwamba, C. Build Custom Directives in Angular 2 [WWW] https://www.codementor.io/christiannwamba/build-custom-directives-in-angular-2-jlqrk7dpw (09.05.2017)

[31] Angular – NPM [WWW] https://www.npmjs.com/package/angular (09.05.2017)

[32] React (Javascript Library) – Wikipedia [WWW] https://en.wikipedia.org/wiki/React_(JavaScript_library) (09.05.2017)

[33] Fedosejev, A. (2015) React.js Essentials. Birmingham : Packt Publishing 18-19

[34] React- A Javascript library for building user interfaces – React [WWW] https://facebook.github.io/react/ (09.05.2017)

[35] ReactJS Overview – Tutorialspoint [WWW] https://www.tutorialspoint.com/reactjs/reactjs_overview.htm (09.05.2017)

[36] Szczeciński, B. How to make AJAX requests in React? [WWW] https://medium.com/@baphemot/how-to-make-ajax-requests-in-react-a6a52bb5a8b1 (09.05.2017)

[37] React – NPM [WWW] https://www.npmjs.com/package/react (09.05.2017)

[38] Introduction – Vue.js [WWW] https://vuejs.org/v2/guide/ (09.05.2017)

[39] Vue – Github [WWW] https://github.com/vuejs/vue (09.05.2017)

[40] Deyne, S. D. Dealing with templates in Vue.js 2.0 [WWW] https://sebastiandedeyne.com/posts/2016/dealing-with-templates-in-vue-20 (09.05.2017)

[41] Vue – NPM [WWW] https://www.npmjs.com/package/vue (09.05.2017)

[42] Ogier, A. Node.js performance 2017: v7.9.0 vs. Hapi, Express.js, Restify and Koa and more [WWW] https://raygun.com/blog/node-js-performance-2017 (09.05.2017)

[43] Krause, S, JS web frameworks benchmark – Round 4. [WWW] http://www.stefankrause.net/wp/?p=316 (10.05.2017)

[44] Arsenault, C. A Comprehensive Overview of WebPack. [WWW] https://www.keycdn.com/blog/webpack/#What-is-Webpack (10.05.2017)

[45] Vue Router – GitHub [WWW] https://github.com/vuejs/vue-router (10.05.2017)

[46] Intro – Vuex [WWW] https://vuex.vuejs.org/en/intro.html (10.05.2017)

[47] Firebase – Firebase [WWW] https://firebase.google.com (10.05.2017)

[48] Firebase – Wikipedia [WWW] https://en.wikipedia.org/wiki/Firebase (10.05.2017)

[49] Vaughan-Nichols, S. J. What is Docker and why is it so darn popular? [WWW] http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/ (10.05.2017)

[50] Docker Cloud – Docker [WWW] https://cloud.docker.com/ (10.05.2017)

[51] Create your first service – Docker [WWW] https://docs.docker.com/docker-cloud/getting-started/your_first_service/ (10.05.2017)

[52] Introduction to Redis – Redis [WWW] https://redis.io/topics/introduction (10.05.2017)

[53] Graph API Overview – Facebook [WWW] https://developers.facebook.com/docs/graph-api/overview (10.05.2017)

[54] Fbgraph – NPM [WWW] https://www.npmjs.com/package/fbgraph (11.05.2017)

[55] Kift events – Kift [WWW] http://www.kift.place (10.05.2017)

[56] Kift events control panel – Kift [WWW] http://www.kift.place:1337 (10.05.2017)

[57] Sublime Text 3 – Sublime Text [WWW] https://www.sublimetext.com/ (10.05.2017)

[58] Git Bash – Git for Windows [WWW] https://git-for-windows.github.io/ (10.05.2017)

[59] Chrome For Desktop – Google [WWW] https://www.google.com/chrome/browser/desktop/ (10.05.2017)

[60] Postman – Postman [WWW] https://www.getpostman.com/ (10.05.2017)

[61] Kift public API – Kift [WWW] http://www.kift.place:1337/api/public/kift (10.05.2017)

[62] Vue Server-side rendering – Vue [WWW] https://vuejs.org/v2/guide/ssr.html (10.05.2017)