TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Aleksei Kadeikin

Ettevõttesisese tarkvara serveripoolse rakenduse täiendamine

Bakalaureusetöö

Juhendaja: Jelena Vendelin

Tallinn 2022

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: Aleksei Kadeikin

18.05.2021

Annotatsioon

Ettevõttesisese tarkvara serveripoolse rakenduse täiendamine

Käesoleva lõputöö eesmärk on parandada ressursside jälgimise süsteemi funktsionaalsust Pipedrive OÜ ettevõttes.

Hetkel puudub Pipedrive OÜ-l võimalus ettevõttesiseselt arendusprojektide jaoks inimressurssi samamoodi planeerida. Iga osakond planeerib ressursse kasutades erinevaid platvorme ning seetõttu puudub ülevaade kogu ettevõtte vabast inimressursist.

Pipedrive OÜ on loonud eraldi tööriista, mis koondab kogu info arendusmeeskondade ja käimasolevate arendusprojektide kohta. Praegu on võimalik näha iga inimese ja nende projektide ajalugu. Ressursside leidmise hõlbustamiseks peame looma ühise keskkonna kogu vajaliku teabega. Teave, näiteks nimekiri projektist, kes praegu osalevad või kavatsevad tulevikus osaleda, kui palju arendajaid on puhkusel ja kui palju on projektideta jne. Selleks luuakse veebirakendus, milles kasutaja näeb jooksvaid projekte ja kogu vajalikku statistikat tulevaste arendusprojektide planeerimiseks. Selles lahenduses saate luua uusi projekte ja teavitada sellest lahendusest otse projekti jaoks sobivaid inimesi (saata kutseid, lisada / eemaldada projekti jne).

Serveriosa ja sellega suhtlemise jaoks tuleb rakendada järgmised lahendused:

1. Uue andmebaasi struktuuri juurutamine või vana optimeerimine, et muuta see funktsionaalsemaks.
2. Juurutada kõik vajalikud päringud andmebaasi serveri kaudu.
3. Andmete taastamiseks vajalikus formaadis olemasolevate optimeerimine ja uute algoritmide lisamine.

Lõputöö on kirjutatud Eesti keeles ning sisaldab teksti 44 leheküljel, 10 peatükki, 8 joonist, 4 tabelit.

Abstract

Upgrading The Server-Side Application Of Internal Software

The aim of this thesis is to improve the functionality of the resource tracking system in Pipedrive OÜ.

At the moment, Pipedrive OÜ[1] does not have the opportunity to plan human resources for development projects in the same way within the company. Each department plans different platforms using resources, and therefore there is no overview of the free human resources of the entire company.

Pipedrive OÜ has created a separate tool that gathers all information about development teams and ongoing development projects. It is now possible to see the history of each person and their projects. To make it easier to find resources, we need to create a common environment with all the necessary information. Information, such as a list of people currently involved or planning to participate in the project, how many developers are on holiday and how many are without projects, etc. To this end, a web application will be created in which users will see current projects and all the necessary statistics for planning future development projects. In this solution, you can create new projects and notify the right people about the project directly (send invitations, add / remove a project, etc.).

The following solutions must be implemented for the server part and communication with it:

1. Implementing a new database structure or optimizing an old one to make it more functional
2. Deploy all necessary queries to the database through the server
3. Optimization of existing and addition of new algorithms in the required format for data recovery.

The thesis is in Estonian and contains 44 pages of text, 10 chapters, 8 figures, 4 tables.

Lühendite ja mõistete sõnastik

AAA	*Arrange Act Assert*
action	*see on lihtne objekt, mis esindab kavatsust muuta olekut*
API	*Application Programming Interface*
CLI	*Command-Line Interface*
CRM	*Customer relationship management*
cron-job	*ajapõhine ülesannete planeerija*
CSV	*Comma-Separated Values*
describe	*see on Jest ja Chai meetod, mis sisaldab ühte või mitut seotud testi*
each	*ntrollvoo avaldus iteratsiooni määramiseks, mis võimaldab koodi korduvalt käivitada Jest raamistikus*
EventMachine::	*see on sündmustepõhine I/O raamatukogu*
for-tsükkel	*kontrollvoo avaldus iteratsiooni määramiseks, mis võimaldab koodi korduvalt käivitada.*
forEach-tsükkel	*kontrollvoo avaldus iteratsiooni määramiseks, mis võimaldab koodi korduvalt käivitada*
Full-Stack	*iseloomustab platvormi, veebisaidi või rakenduse loomiseks kasutatavate tarkvaralahenduste ja tehnoloogiate teadmiste kogumit*
G&A	*General and Administrative*
Git	*hajutatud versioonihaldussüsteem, mis on loodud mis tahes projektide kiireks ja tõhusaks töötlemiseks.*
GNU	*Unixi sarnane operatsioonisüsteem*
HTTP	*Hypertext Transfer Protocol*
I/O	*Operating System*
ID	*Identity Document*
it	*testplokk*
jQuery	*JavaScripti raamistik, mis on loodud HTML DOM-i puu läbimise ja manipuleerimise hõlbustamiseks*
JSON	*JavaScript Object Notation*
lowerCamelCase	*nimetamiskokkulepe, mille puhul nimi moodustatakse mitmest sõnast, mis on liidetud üheks sõnaks, kusjuures iga sõna esimene täht (välja arvatud esimene) kirjutatakse nime moodustavas uues sõnas suurtähtedega.*
MVP	*Minimum Viable Product*
OS	*Operating System*
PaaS	*Platform as a service*
POST	*HTTP-meetod, mis on loodud määratud ressursist serverisse suure hulga andmete saatmiseks.*
SPA	*Single-Page Application*
URL	*Uniform Resource Locator*
where-klause	*kasutatakse tingimuse määramiseks andmete toomisel ühest tabelist või mitme tabeliga ühendamisel.*
XML	*eXtensible Markup Language*

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Antud peatükis antakse lugejale lühiülevaade teostatud lahendusest, selle eesmärgist ning probleemist. Samuti kirjeldatakse üldsõnaliselt loodud funktsionaalsust ning meeskondliku lõputöö edasist ülesehitust.

1.1 Üldine taust ja projekti lühikirjeldus

Pipedrive™ Estonia (edaspidi Pipedrive) on esimene müügiprofessionaalide loodud CRM-i platvorm müügiprofessionaalidele[1]. Pipedrive'i visioon on saada maailma juhtivaks CRM-lahenduste pakkujaks ning missiooniks on pakkuda kõige mugavamaid, praktilisemaid ja informatiivsemaid tööriistu müügi parandamiseks.

Väljatöötatud projekt on ITB1706 *„Infosüsteemide arendamise meeskonnaprojekt: tellimus (2021/2022 sügis)“* aine raames käivitatud ettevõttes läbiviidavate projektide haldamise, loomise, muutmise lahenduse edasiarendus. Lahendus ettevõtte projektide haldamiseks, loomiseks, muutmiseks on täiendus olemasolevale Pipedrive'i kasutatavale rakendusele ning mõeldud eelkõige toote- ja missioonijuhtidele.

See lahendus:

1. aitab anda ülevaate varasemast ja praegusest arendusüksuste tööjõustatistikast;
2. annab võimaluse luua uus projekt ja valida selle elluviimiseks parim aeg;
3. võimaldab jälgida arendusüksuse käimasolevaid ja tulevasi projekte ning liikmeid koos nendega seotud infoga konkreetsel perioodil.

Lahendus sobib nii väikestele kui suurtele meeskondadele või ettevõtetele.

1.2 Probleem

Infohaldussüsteemide, eriti arendatud süsteemide optimaalseks toimimiseks ja hooldamiseks on vaja inim-, materiaalseid ja ajaressursse. Praegu on tegevuste automatiseerimise probleem muutunud oluliseks kõikide ettevõtete jaoks, olenemata nende suurusest ja tegevuse liigist.

Uue funktsionaalsuse ehitamiseks ja edukaks juurutamiseks on vajalik tööriista ja ettevõtte üksikasjalik uurimine, et tuvastada olemasolevad probleemid, analüüsida nende lahendamise võimalust ja otstarbekust oma tegevuse täieliku või osalise automatiseerimise abil.

Manuaalse andmete sisestamise probleemi lahendamine on tähtis, et hoida kokku töötajate aega ja mugavdada nende tööd. Tänaseks on tekkinud arusaam struktureerimata teabe automatiseerimise, registreerimise ja töötlemise vajadusest, kuna selle maht kasvab iga aastaga ja käsitsi töötlemine on ajamahukas.

Sellest tulenevalt on uurimistöö objektiks CRM ettevõte.

See lahendus tagab parema tööjõu- ja aja planeerimise ning vähendab manuaalset tööd uute projektide kavandamisel ja olemasolevate kohandamisel. Ettevõttel on võimalus projekte edaspidi täpsemalt ja mugavamalt planeerida minimaalsete kahjudega. Samas võimaldab tööjõuhalduse lahendus saada kogu vajaliku info ühest kohast, samuti hoiab ära töötajate ülekoormamise ja tööjõupuuduse.

Peamised kasutajad ja sihtrühmad on:

- Tootejuht - kes on ettevõttes uute projektide peamine initsiaator;
- Projektijuht - kes tagab projekti kulgemise, töökorralduse ja kõik vajalikud ressursid koordineeritud tööks projekti käigus.

Projekti eesmärk on luua ressursihaldussüsteem, millel on võimalus luua, muuta ning jälgida uusi ja olemasolevaid projekte.

1.3 Eesmärk

Pipedrive OÜ on loonud eraldi tööriista, mis koondab kogu info arendusmeeskondade ja käimasolevate arendusprojektide kohta. Praegu on võimalik näha iga inimese ja nende projektide ajalugu. Ressursside leidmise hõlbustamiseks tuleb luua ühine keskkond kogu vajaliku teabega. Teabeks on nimekiri projektidest, töötajatest, kes praegu osalevad või kavatsevad tulevikus osaleda, puhkusel olevate arendajate arv ja töötajate arv, kellel puuduvad projektid jne. Selleks luuakse veebirakendus, milles kasutajad näeb jooksvaid projekte ja kogu vajalikku statistikat tulevaste arendusprojektide planeerimiseks. Selles lahenduses saab luua uusi projekte ja kutsuda loodavasse projekti arendajaid neile kutse saates Slack[2] keskkonda.

Praegu kasutatakse Confluence[3] tööriista uute projektide registreerimiseks, mis on ühendatud veebirakenduse Mission Tool'iga.

Serveriosa ja sellega suhtlemise jaoks tuleb rakendada järgmised lahendused:

1. Uue andmebaasi struktuuri juurutamine või vana optimeerimine, et muuta see funktsionaalsemaks.

2. Juurutada kõik vajalikud päringud andmebaasi serveri kaudu.

3. Andmete taastamiseks vajalikus formaadis olemasolevate optimeerimine ja uute algoritmide lisamine

1.4 Lühiülevaade teostatud funktsionaalsusest

Veebirakenduses on võimalik saada andmeid nii Confluence'ist kui ka andmebaasist. Neisse salvestatakse kogu info kõigi projektide ja töötajate kohta, mille alusel koostatakse statistika kalendri, käimasolevate projektide ja nende ajaraamidega, aga ka arendusüksuses olevate inimeste ja nende töökohtade kohta. Kogu eelnevat infot arvestatakse iga nädala kohta, kolme aasta perioodil. Päringute kaudu saab määrata, millist infot tuleb veebilehele tagastada, küsida konkreetse perioodi statistikat ja vajadusel saada lisainfot.

Samuti loodi lahendus, mille eesmärk oli ühendada veebirakendus Missions Tool ettevõttes töötajate vaheliseks suhtluseks kasutatava rakendusega – Slack. Projekti raames tehti seda selleks, et muuta mugavaks inimeste kutsumine projekti ja hoida aega kokku igale inimesele eraldi kirjutamisega. Samuti on võimalik jälgida, kes on juba uude projekti kutsutud. Varem tuli kõike seda teha manuaalselt.

Samamoodi realiseeriti kõik vajalikud päringud andmebaasi, mida oli vaja ülaltoodud funktsionaalsuste rakendamiseks.

Kuna need eesmärgid eeldasid nii uute tabelite lisamist andmebaasi kui ka olemasolevate muutmist, siis seda ka selle töö raames tehti.

Kogu loodud funktsionaalsus on lõppkasutajale mugav ning tagab ka suure hulga teabe vastuvõtmise ja lisamise ühes kohas minimaalse ajakuluga selle hankimiseks.

2 Veebirakendus Mission Tool

2.1 Mission Tool

Seda Pipedrive'i organisatsiooni poolt välja töötatud tööriista kasutatakse selleks, et saada ülevaade ettevõttes käimas olevatest projektidest, arendusüksustest ja nende all olevate inimeste info kättesaamiseks. Lehtedel kuvatav sisu on enamasti staatiline, välja arvatud jaotised „Toote omadused" ja „Konkurendid", mille andmed laaditakse Airtable'st [4].

Toodet kasutatakse ettevõttes laialdaselt ja aktiivselt arendajate seas, kuna see hõlbustab oluliselt uute projektide planeerimise protsessi ning sisaldab ka suurt hulka teavet, mis kasutajale automaatselt omastatakse.

See tööriist on kätte saadav ainult ettevõttes Pipedrive OÜ ja on saadaval ainult ettevõtte siseseks kasutamiseks.

2.2 Mission Tool enne projekti

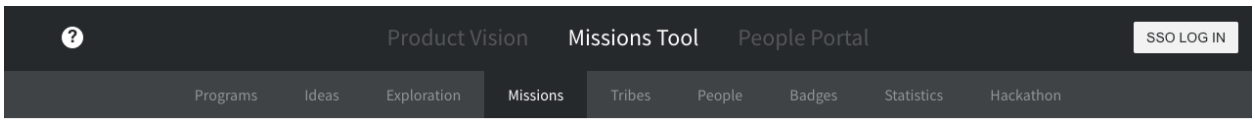
Algselt oli selle tööriista põhieesmärk jälgida ja anda teavet ettevõttes käimasolevate projektide kohta, samuti aidata uusi projekte käivitada ja anda teavet selle kohta, mis Pipedrive ettevõtte arendusüksustes ühel või teisel ajal toimus. Kuid aja jooksul on selle funktsionaalsus laienenud.

Selle tööriista iga funktsionaalsuse jaoks on loodud eraldi leht, millel on erinev teave ja funktsionaalsus.

Lehed ja nende funktsionaalsus:

- Programmid – vaata kõiki programme, milles ettevõte osaleb;
- Uurimine – teave selle kohta, millised uuringud ettevõttes hetkel toimuvad;
- Missioonid – rubriik, kus saate tutvuda kõikide projektidega (nii pooleliolevate kui ka arhiveeritud), mis on ettevõttes korraldatud. Siit võib iga ettevõtte arendaja leida teda huvitava projekti ja jätta avalduse selles osalemiseks;
- Arendusüksused – rubriik, kus saate tutvuda kogu olemasoleva teabega, mis on seotud ettevõtte kõikide arendusmeeskondadega;

- Inimesed – jaotis, mis annab teavet ettevõtte kõigi töötajate, nende töökohtade, projektide, ja muu teabe kohta. Töötajad võib jagada järgmistesse kategooriatesse:
    - Arendajad (*Engineering*);
    - Tootehaldus (*Product Management*);
    - Tehnilised toimingud (*Engineering Operations*);
    - Äriarendus (*Business Development*);
    - G&A;
    - Turundus (*Marketing*);
    - Müük (*Sales*);
    - Toetus (*Support*);
    - Kliendiabi (*Customer Success*);
    - Tehniline platvorm (*Engineering Platform*).
- Märgid – rubriik, mis tutvustab kõiki ettevõtte töötajate saavutusi;
- Statistika – selles jaotises saate Grafana[5] teenust kasutades vaadata üksikasjalikku statistikat selliste rubriikide kohta nagu projekti üldine statistika, projekti arendajate arv ja palju muud. Jaotis pakub ka võimalust jälgida projekte, meeskondi, vabu arendajaid ja palju muud.



Joonis 1. Mission Tooli veebilehe navigeerimisriba

Kõik ettevõttes toimuvad projektid ja ka täielik info nende kohta sisalduvad Confluence keskkonnas. Samuti on ülaltoodud info salvestatud andmebaasi, mis sisaldab ka andmeid ettevõtte töötajate kohta.

Andmed töötajate ja projekti kohta sisestatakse andmebaasi *“cron-job”* kaudu, mis teatud ajal täiendavad andmebaasi sõltuvalt nende andmete peamistes salvestusressurssides tehtud muudatustest:

- Projektide info – Confluence
- Töötajate info – BambooHR[6]

3 Metoodika

See peatükk annab lühiülevaate loodavast objektist, arenduses tekkinud probleemidest, kasutatud tööriistadest ja tehnoloogiatest ning tööprotsessist. Lisaks tuuakse välja kasutusjuhtude abil töö nõuded.

3.1 Objekti detailne kirjeldus

Selle projekti käigus välja töötatud lahendus on täiendus juba olemasolevale lahendusele – Missions Tool. Missions Tool on Pipedrive'i poolt välja töötatud veebirakendus, mida kasutatakse ettevõte siseselt käimasolevate projektide haldamiseks, töötajate tööhõive jälgimiseks, statistika hankimiseks ja eelmainitu kohta täpsema info hankimiseks. Seda rakendust käsitletakse üksikasjalikumalt 2.1 peatükis.

Selles töös väljatöötatud projekti alustas 6-liikmeline meeskond: Lii Saluvere, Annabel Matkur, Mariliis Sinivee, Sergei Vassiljev, Aleksei Kadeikin, Reia Rõõmus – kes kõik on äriinfotehnoloogia 3. Kursuse tudengid. Projekt sai alguse õppeaine ITB1706 "*Infosüsteemide arendamise meeskonnaprojekt: tellimus (2021/2022 sügis)*" raames, kus ülesandeks oli selle projekti MVP väljatöötamine. Pärast seda otsustati antud projektiga jätkata lõputööna, mille eesmärgiks oli juurutada kogu vajalik funktsionaalsus antud rakenduse mugavamaks kasutamiseks.

Aine ITB1706 raames võttis projekt oodatust kauem aega. Projekti tarneaja pikenemise põhjuseks oli meie meeskonna poolt välja töötatud olemasolevate lahenduste uurimisele ja uute lahenduste integreerimisele kulunud aeg. See andis aga suure hulga teadmisi projekti elluviimiseks lõputöö raames.

"Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine"[7] töö raames selgitati välja parim lahendus uute projektide loomiseks ja inimeste kutsumiseks ning loodi prototüüp. Lisaks selgitati välja andmebaasis puuduvad tabelid ja väljad uute funktsionaalsuste loomiseks. Lõputöö autori ülesandeks oli realiseerida vajalikud muudatused andmebaasis ja luua vajalikud tagarakenduse ühendused.

Töö alguses pakkuti arendajatele kõik tööks vajalikud komponendid:

- Juurdepääs kõigile vajalikele materjalidele ja arenduskeskkondadele.

- Koolitused ja kursused – Pipedrive'is vajalike tehnoloogiate ja protsessidega tutvumiseks on korraldatud sisekursused.
- Mentorid – arendajatele pakkusid tuge ettevõtte töötajad, kes aitasid arendusprotsessi ajal tekkivate probleemidega.
- Seadmed tööks – igal praktikandil on tööks oma personaalarvuti.
- Tööruum – igal praktikandil oli oma töölaud kõige mugavaks igapäevatööks vajalikuga.

Lahendus töötati välja nii Vue.js kasutajaliidese arendusraamistikku kui ka avatud lähtekoodiga Node.js serveri arendusraamistikku kasutades. Nii otsustati, kuna rakendus töötas algselt nende raamistike peal ja nendega oli arendajatel varasem kogemus veebirakenduste arendamisel.

3.2 Arendamise käigus ilmnenud probleemid

Esimene probleem, millega selle projekti algusest peale kokku puutusime, oli Vue.js kasutajaliidese versioon – 2.6.12, samas kui projekti kirjutamise ajal oli olemas juba Vue.js versioon 3.2.33. See on probleem, kuna viimases versioonis on täiustatud rakenduse kiirust, komponentidevahelise interaktsiooni kergeid meetodeid ning arendajasõbralikumat tööriistakomplekti. Kõik need täiustused võimaldavad saada kiirema, sõbralikuma ja parema toote. Selle probleemi lahendus, rakenduse tavaline värskendamine uusimale versioonile, on üsna lihtne, kuid ainult esmapilgul, sest selle probleemi lahenduse kasutamisel peab kogu rakenduse kohandama uue versiooniga, mille juurutamine võtaks liiga palju aega ja mis samuti ei kuulu selle projekti ülesannete hulka. Samuti oli rakenduse arenduse jätkamise põhjuseks versioonil 2.6.12 senine eesmärk kasutada rakenduses võimalikult palju olemasolevaid komponente ja lahendusi ning uuele versioonile üleminek võib kaasa aidata nende ebaõigele tööle, rikkele või neist täielikult loobumist, kuna uus versioon ei suuda nende komponentidega suhelda [8].

Teine probleem, millega selle projekti väljatöötamisel kokku puutusime, oli asjaolu, et see rakendus on 3-5 aastat vana, samuti uute arenduste puudumine sama perioodi jooksul. Nii pikk tegevuse puudumine rakenduses aitas kaasa üksikasjaliku teabe puudumisele selle rakenduse kohta, suhtluse puudumine selle rakenduse algse omaniku ja sellega seotud arendajatega. Aja jooksul on ettevõte teinud palju muudatusi ja ümberkorraldusi, mille tõttu on info ununenud või või kaduma läinud.

3.3 Tööriistade kirjeldus

Kogu rakendus on täielikult realiseeritud JavaScript'i programmeerimiskeele abil, kasutades avatud teekide kujul abi tööriistu:

- dateFNS – pakub kõikehõlmavaid funktsioone kuupäeva vormindamiseks ja manipuleerimiseks;
- Axios – Promise-põhine HTTP-klient brauseri ja Node.js jaoks;
- Joi – täidab ruuterilt tulevate sisendandmete valideerimise funktsiooni;
- Knex – SQL päringu koostaja, mis pakub hõlpsasti kasutatavat liidest andmebaasile ja selle andmetele juurdepääsuks ja nendega tegutsemiseks;
- Koa – sellest teegist kasutatakse ainult koa-router, mis tagab ruuteri funktsionaalsuse serveri osas;
- Lodash – pakub funktsionaalse programmeerimise kasutamisel abifunktsioone JavaScript'i programmeerimiskeelega töötamiseks;
- MySql – peamine rakenduse arendamisel kasutatav andmebaasihaldussüsteem;
- Node-Cron – on GNU *crontab'*il põhinev JavaScript'i ülesannete ajakava Node.js jaoks, mis võimaldab ajastada ülesandeid failis Node.js, kasutades täielikku *crontab'*i süntaksit;
- vue-router – kasutatakse üheleheliste rakenduste (*SPA*) marsruutimise konfigureerimiseks ja juurutamiseks;
- Vuex – täidab rakenduse oleku haldamise rolli ja pakub ka suurt hulka funktsioone Vue.js raamistikus põhinevate rakenduste jaoks. Rakenduses kasutatakse peamiselt kõigi rakenduse komponentide tsentraliseeritud hoidlana;
- Babel – JavaScript'i kompilaator;
- Eslint – täidab kirjutatud koodi kontrollimise funktsiooni – hoolitseb selle eest, et kood oleks ühtses stiilis, aitab vältida vigu, suudab paljusid vigu iseseisvalt parandada, töötab paremini kui ükski teine teek koos Vue.js'iga;
- Mocha – raamistik funktsionaalsete testide kirjutamiseks ja kontrollimiseks;

- Jest – raamistik üksiktestide kirjutamiseks ja kontrollimiseks;
- Nodemon – on käsurea liidese (*CLI*) utiliit, mis jälgib Node'i rakenduse failisüsteemi ja vajadusel taaskäivitab protsessi automaatselt.

Kogu rakendus töötati välja Visual Studio Code'is[9], millel on Microsoft'i litsents. Selle konkreetse redaktori valikut mõjutas selle ettevõtte poliitika, kus kogu töö käesoleva lõputöö raames tehti, kuna absoluutselt kõik ettevõtte töötajad, kes ühel või teisel viisil mõjutavad koodi kirjutamist või testimist, peavad kasutama ülaltoodud redaktorit. Ettevõte põhjendas seda valikut asjaoluga, et tegemist on „läbipaistva" rakendusega, mis ei kujuta endast mingit ohtu ning kuna peaaegu kogu ettevõtte arendus toimub JavaScript'i programmeerimiskeele abil, mis on kõige paremini kohandatud tööks toimetaja kohal, langes valik just ülaltoodud redaktori suunas.

GitHub, tarkvaraarenduse ja *Git'*i kasutava versioonikontrolli Interneti-majutusteenuse pakkuja, toimis peamise koodi hoidlana ja ka äsja kirjutatud koodi kontrollerina [10].

Docker on platvormi-teenusena (*PaaS*) toodete komplekt, mis kasutab *OS'*i tasemel virtualiseerimist, et tarnida tarkvara pakettides, mida nimetatakse konteineriteks – just seda rakendust kasutati rakenduses kasutatava andmebaasi konteinerite loomiseks ja käitamiseks [11].

Peamise arenduskeskkonnana selle konkreetse projekti raames toimis Node.js, millest tuleb täpsemalt juttu järgmises osas.

Jira platvormi kasutati tööde planeerimiseks ja jälgimiseks, samuti ülesannete koostamiseks [12].

3.4 Tehnoloogiate kirjeldus

See peatükk sisaldab üksikasjalikumat teavet peamiste tehnoloogiate kohta, mida selle rakenduse väljatöötamisel kasutati.

3.4.1 Back-End

Selle rakenduse peamine arenduskeskkond on Node.js raamistik, mis on sündmustepõhise asünkroonse JavaScript'i käituskeskkonnana loodud skaleeritavate võrgurakenduste loomiseks. See raamistik valiti, kuna see erineb tänapäevasest enamlevinud samaaegsusmudelist, mis kasutab *OS*'i lõime. Voolupõhine võrk on suhteliselt ebaefektiivne ja väga raskesti kasutatav.

Samuti on Node.js'il eelis protsessi pidama jäämise korral, kuna Node.js'il neid pole. Peaaegu ükski Node.js'i funktsioon ei teosta otse I/O'd, seega protsess ei blokeeri kunagi, välja arvatud

juhul, kui sisend-/väljundit tehakse Node.js'i standardteegi sünkroonsete meetoditega. Kuna miski ei blokeeri, on Node.js'is skaleeritavate süsteemide arendamine väga mõttekas.

Node.js on disainilt sarnane ja mõjutatud sellistest süsteemidest nagu Ruby's Event Machine ja Python's Twisted. Node.js viib sündmuste mudeli veidi kaugemale. See esitleb sündmusetsüklit käitusaegse konstruktsioonina, mitte raamatukoguna. Teistes süsteemides on sündmusetsükli käivitamiseks alati blokeerimiskõne. Tavaliselt määratletakse käitumine skripti alguses tagasihelistamisega ja lõpus käivitatakse server blokeeriva kõnega, näiteks *EventMachine::run()*. Node.js'il pole sellist väljakutset sündmuse tsükli käivitamiseks. Node.js siseneb lihtsalt sündmuse tsüklisse pärast sisestus skripti täitmist. Node.js väljub sündmuse tsüklist, kui käivitamiseks pole enam tagasihelistusi. See käitumine sarnaneb brauseri JavaScript'iga – sündmuse silmus on kasutaja eest peidetud.

HTTP on Node.js-s esmaklassiline osaleja, mis on loodud voogesitust ja madalat latentsust silmas pidades. Seetõttu sobib Node.js hästi veebiteegi või raamistiku jaoks.

See, et Node.js on loodud ilma lõimedeta, ei tähenda, et rakenduskeskkonnas on võimatu ära kasutada mitut tuuma. *Child_process.fork()* API abil saab luua alamprotsesse, millega on lihtne suhelda. Sama liidese põhjal on ehitatud klastri moodul, mis võimaldab jagada pistikupesasid protsesside vahel, et tagada koormuse tasakaalustamine süsteemi tuumade vahel [13].

3.4.2 Front-End

Vue.js raamistik võeti kohandatud rakenduse arendamise aluseks.
Nagu iga hea raamistik, kasvab ja areneb ka Vue.js, nii et see pakub algusest peale rohkem funktsioone, kui lubab. Praegu pakub see lihtsat viisi pistikprogrammide ühendamiseks ja loomiseks, segalahenduste kirjutamiseks ja kasutamiseks ning üldiselt kohandatud käitumise lisamiseks. Vue-d saab kasutada paindlikult ja see on rakenduste struktureerimisel nii erapooletu, et seda võib kindlasti pidada raamistikuks, mis toetab keeruliste veebirakenduste loomist lõpuni [14].

Kuna antud töö raames oli peamiseks ülesandeks selle rakenduse serveriosa, samuti vajalike funktsioonide ja algoritmide väljatöötamine, siis selles osas kasutajaliidese arendusse sügavamat sukeldumist ei toimu ja põhimõtteliselt selles töös. Teavet selle osa kohta on *"Ettevõttesisese tarkvara kasutajaliidese täiendamine"*[15] tööst.

3.5 Tööprotsessi kirjeldus

Tööprotsess jagunes kolmeks põhiliseks osaks ettevalmistamine, arendus ning vahetulemuste valideerimine. Nendest lähemalt on kirjas järgnevates kolmes alampeatükis.

3.5.1 Ettevalmistus

Peale meeskonnaprojekti kaitsmist ITB1706 aine raames toimus meil kohtumine ettevõttes projektijuhiga, samuti insenerijuhiga. Seekordsel kohtumisel võeti kokku tehtud töö tulemused, mis kujunesid juhi hinnangul positiivseks – saavutati palju eesmärke, märgati projektis osalevate inimeste edusamme ning saavutati ka tulemus. Isegi kui võtta arvesse asjaolu, et tööd ei jõudnud lõpuni, jäi firma rahule. Seejärel selgitati välja edasised tööfaasid, püstitati uued ülesanded ning arutati läbi kõik lõputööga seotud aspektid, mis võimaldas ITB1706 aine raames alustatud projektiga probleemideta jätkata tööd lõputööna.

Pärast eelpool mainitud kohtumist algas edasise töö planeerimine. Kuna selle projekti ülesandeks on juurutada kõik *"Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine"* projekti elluviimiseks vajalik, siis enne serveriosa kallal töö alustamist tuleb saada infot selle kohta, milline saab rakendus olema, kuidas seda korraldatakse, mida on vaja ja mida mitte, milliste asjadega peame töötama ja palju muud – seda kõike tegid *"Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine"* projekti raames Lii Saluvere, Annabel Matkur. Kohe oli aga teada mitmeid detaile, mille jaoks oli vaja teha lokaalne analüüs – millest töö alguse sai.

Kui *"Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine"* töö raames tehti analüüs ja loodi tulevase rakenduse prototüüp, tekkis võimalus teha veel üks lokaalne analüüs, kuid seekord oli ajendatud valmisandmetest, mida on vaja täisväärtuslikuks arengu käivitamiseks.

Kohe peale lokaalse analüüsi teostamist said vastused kõik prototüübiga seotud küsimused, saadi edasiseks tööks vajalikuks osutunud teadmised ning asuti planeerima töövoogu. Peatükis 2.2 mainitud Jira ja GitHub süsteemid olid peamisteks töövahenditeks edasise töö planeerimisel ja töö enda käigus. Nende kasutamist käsitletakse järgmises jaotises.

Kuna see projekt on jätk ITB1706 raames alustatud projektile, oli GitHub, mis toimib selle projekti käigus tehtud töö peamise hoidlana, juba täielikult konfigureeritud ja töövalmis. Samuti olid kõik meeskonnaliikmed nii käesoleva lõputöö kui ka kogu projekti raames hästi kursis *Git* süsteemi

toimimisega, mis hõlbustas töövoogu elementaarsete küsimuste puudumise kõigi arendatud projektis osalejate seas.

Samuti ei olnud probleemiks Jira ressurssi kasutades edasise töö organiseerimine, ülesannete püstitamine, tähtaegade seadmine, eesmärkide selge väljatoomine, kuna seda kasutati ka projekti väljatöötamisel ITB1706 aine osana.

3.5.2 Arendus

Töövoo esimene ülesanne oli selgelt ja asjatundlikult korraldada *"Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine"* töö raames väljatöötatud prototüübi analüüsi käigus kindlaks määratud ülesannete järjekord ja ulatus. Analüüsiga tehti kindlaks kõik selle projektiga töötamiseks vajalikud komponendid – töökoht, töö skeem, vajalikud teadmised, tarkvara, ajaraam ja muud väiksemad aspektid. Selle töö raames analüüsiti ka väljatöötatud prototüüpi – määrati ülesannete järjekord, nende keerukus, teostusmeetodid, peamised kasutusjuhud ja võimalikud raskused mis tahes ülesande täitmisel.

Otsustati määrata põhiülesannete rühmad, mida täidetakse ükshaaval või mitu korraga, kui selline võimalus tekib samaaegse rakendamise ja/või ajutise reservi poolelt. Samuti otsustati, et kogu ühe ülesannete rühmaga seotud ülesannete kompleks planeeritakse selgemalt enne selle ülesannete rühma kallal töötamist.

Tööd tehti peamiselt kontorist, kuna see oli mugav töökeskkond, mis võimaldas ilma probleemideta ja raskusteta, nagu näiteks probleemid internetiga, seda projekti arendada, aga ka olla ühenduses teiste meeskondadega, kes samuti selle projekti väljatöötamise ja analüüsiga tegelesid, mis on iga projekti eduka arendamise juures üsna oluline punkt. Juhul, kui sooviti kontorist eemal töötada, kasutati suhtluseks sidekanaleid nagu Slack ja Zoom[16], mida ettevõte kasutas.

Samuti toimus ülaltoodud suhtluskanalite kaudu pidev suhtlus mentoritega, kes vastasid alati meeskonna küsimustele.

Selle projekti arendus sai alguse lahenduse juurutamisest, mis pakub kõiki statistikatabeli andmeid – see tabel on selle projekti võtmeobjekt, kuna just see annab kogu vajaliku info tuleviku planeerimiseks ja käimasolevate projektide jälgimiseks. See lahendus põhines juba olemasoleval lahendusel, mis andis teavet ainult käimasolevate projektide kohta. See on kalender, mis sisaldab teavet käimasolevate projekte, nende kuupäevade, osalejate ja muude väiksemate andmete kohta.

See valiti just seetõttu, et projekti ülesandeks oli viia ellu lõppkasutajale sõbralik lahendus, mis see ka on, kuna see on ettevõttes kasutusel olnud juba mitu aastat ning see suudab kõige täpsemalt edastada vajalikku infot lõppkasutaja. See otsus tehti tootejuhtide projekti raames probleemi lahendamiseks parima variandi väljaselgitamise käigus. Ülesandeks oli välja töötada lahendus kogu kalendri täitmiseks vajaliku info tagastamiseks.

Ülaltoodud lahenduse väljatöötamine osutus üsna keeruliseks, kuna töö tehti suure hulga erinevatest ressurssidest ja andmebaasi tabelitest pärit andmetega. Realiseerimisel kasutati juba olemasolevat API päringut, mis andis küll kogu vajaliku info, kuid andis selle iga päeva vormingus, mis ei vastanud prototüübis kehtestatud kriteeriumitele – andmed tuleks esitada nädalavormingus. Seetõttu otsustati seda täiendada funktsionaalsusega, mis võimaldas neid andmeid sobivas vormingus hankida. Pärast ülaltoodud muudatuste rakendamist alustati kalendrile arusaadava loogika väljatöötamisega, kuna kalendris kasutatav formaat ei ühti päringult tagastatud vorminguga:

1. päring tagastas ainult andmed, samuti ka tulevaste nädalate statistika arvutamise loogika arendamine;
2. päring tagastas andmed ainult möödunud perioodide kohta.

Arendusse võeti järgmine ülesanne, mis on seotud uue projekti loomise loogika rakendamisega. Sellel loogikal on kaks osa:

1. Confluence keskkonnas uue projekti loomine;

2. Uue projekti kohta info lisamine andmebaasi.

Ülesanne jaotati arendajate vahel kaheks. Selle ülesande täitmisel kasutati projekti osi, mida ei olnud varem selle projekti ja teemaga ITB1706 töötamise käigus välja töötatud projekti arenduse raames kasutatud. See asjaolu tõi rakendamisel kaasa mõningaid raskusi ja viivitusi, mis olid tingitud ajaraiskamisest tööriista ja materjali uurimiseks, millega oli vaja töötada, kuid lõpuks õnnestus siiski välja töötada selle osaga seotud tööloogika.

Järgmiseks ülesandeks oli ülaltoodud lahenduse ühendamine kasutajaliidesega, et luua uus projekt, mis töötati välja *“Ettevõttesisese tarkvara kasutajaliidese täiendamine”* töö raames. Kasutajaliidese ühendus serveriga realiseeriti API päringu abil serverile, mis tegi andmebaasi POST päringu koos kõigi saadud andmetega, mille tulemusena tekkis andmebaasis uus kirje – soovitud tulemus.

Viimase ülesandena realiseeriti võimalus kutsuda osalejaid projekti, teavitades neid automaatselt ettevõtte suhtluskeskkonnas – Slack'is. Selle ülesande rakendamise skeem on üsna sarnane uue projekti loomise loogika rakendamise skeemiga – andmebaasis uue kirje loomine, aga ka kolmanda teenusega suhtlemine, kuid kui projektitabelid andmebaasis olid juba olemas, siis selle ülesande hulka kuulus ka andmebaasi uue tabeli loomine, kuhu salvestatakse kõik realiseeritud viipad.

Samuti töötati kõigi ülaltoodud rakenduste jaoks välja nii üksuse kui ka funktsionaalsed testid. Neid kirjeldatakse üksikasjalikumalt 5. Peatükis.

3.5.3 Vahetulemuste valideerimine

Kaks korda nädalas toimusid kogu meeskonna koosolekud, et teha kokkuvõte tehtud tööst, teavitada kõiki meeskonnaliikmeid probleemidest/lahendustest ning otsustada edasised tegevused. Nendel koosolekutel osalesid ainult need meeskonnaliikmed, kes sel hetkel kontoris olid. Harvadel juhtudel, näiteks siis, kui kontor oli kohalolekupiirangute tõttu suletud, peeti koosolekuid veebis, kasutades veebipõhist rühmakoosolekute rakendust Zoom või kirjaliku ettekande meetodil, näiteks kui aruteluteemasid on vähe.

Sellistel kohtumistel just selle projekti raames mentorid ei osalenud, kuna leiti, et nende roll selles projektis oli abistamine, mitte aga täielik osalemine arendusprotsessis.

Mentorid kontrollisid ka kõiki selle projekti arendusmeeskonna liikmete poolt kavandatud muudatusi.

3.6 Peamised kasutusjuhud

Selles tööosas tutvustatakse väljatöötatud rakenduse kasutamise peamisi võimalusi, mis *"Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine"* töö käigus välja määrati.

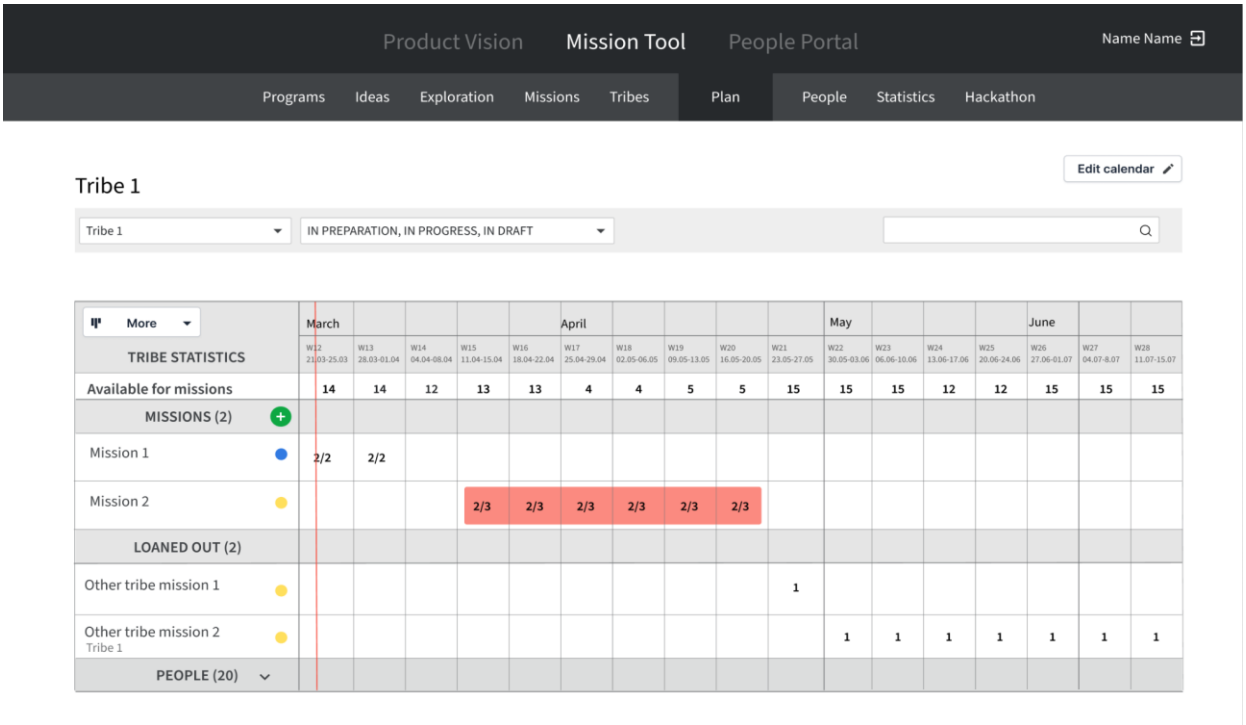
Kasutusjuhtude tekstikirjeldustesse ei ole lisatud tegutsejaid, sest kõigil ettevõtte töötajatel on õigused ja võimalused neid toiminguid rakenduses teha.

3.6.1 Arendusüksuse statistika vaatamine

Nimetus	Arendusüksuse statistika vaatamine

Eesmärk	Kasutaja saab vaadata arendusüksuse statistikat (arendusüksuses töötavate arendajate koguarvu, mitut arendajat on vaja igapäevaste ülesannetega tegelema, kui mitu arendajat tegelikult igapäevaste ülesannetega tegeleb, mitut arendajat on kokku projektidesse vaja, mitu arendajat on süsteemis märkinud, et osalevad konkreetses projektis, mitu vaba kohta veel projektides on, mitu arendajat teistest arendusüksustest konkreetse arendusüksuse projektides osaleb ning mitu arendajat viibib puhkusel) ning filtreerida, millist statistikat arendusüksuse kohta soovitakse nädalate lõikes näha.
Tegevuse kirjeldus	Kasutaja navigeerib *"Planning"* nime all olevale kindla meeskonna planeerimislehele, kus on väljas kogu valitud meeskonnaga seotud statistika ning valitud meeskonnas toimuvad missioonid. Statistikat jaotatakse kirjelduses määratud nimetuse all. URL: "planning/tribe/:id"

Tabel 1. Arendusüksuse statistika vaatamine



Joonis 2. Arendusüksuse statistika vaade prototüüpist [7]

3.6.2 Projekti detailse sisu vaatamine

Nimetus	Projekti detailse sisu vaatamine
Eesmärk	Kasutaja saab vaadata detailset infot projekti kohta sh projekti nime, asukohta, toimumisaega, teemat ning osalejaid. Samuti näeb kasutaja viidet, mis viib projektiga seotud Confluence'i lehele.
Tegevuse kirjeldus	Kasutaja navigeerib *"Planning"* nime all olevale kindla meeskonna planeerimislehele, kus on väljas kogu valitud meeskonnaga seotud statistika ning valitud meeskonnas toimuvad missioonid. Seal osutab ta

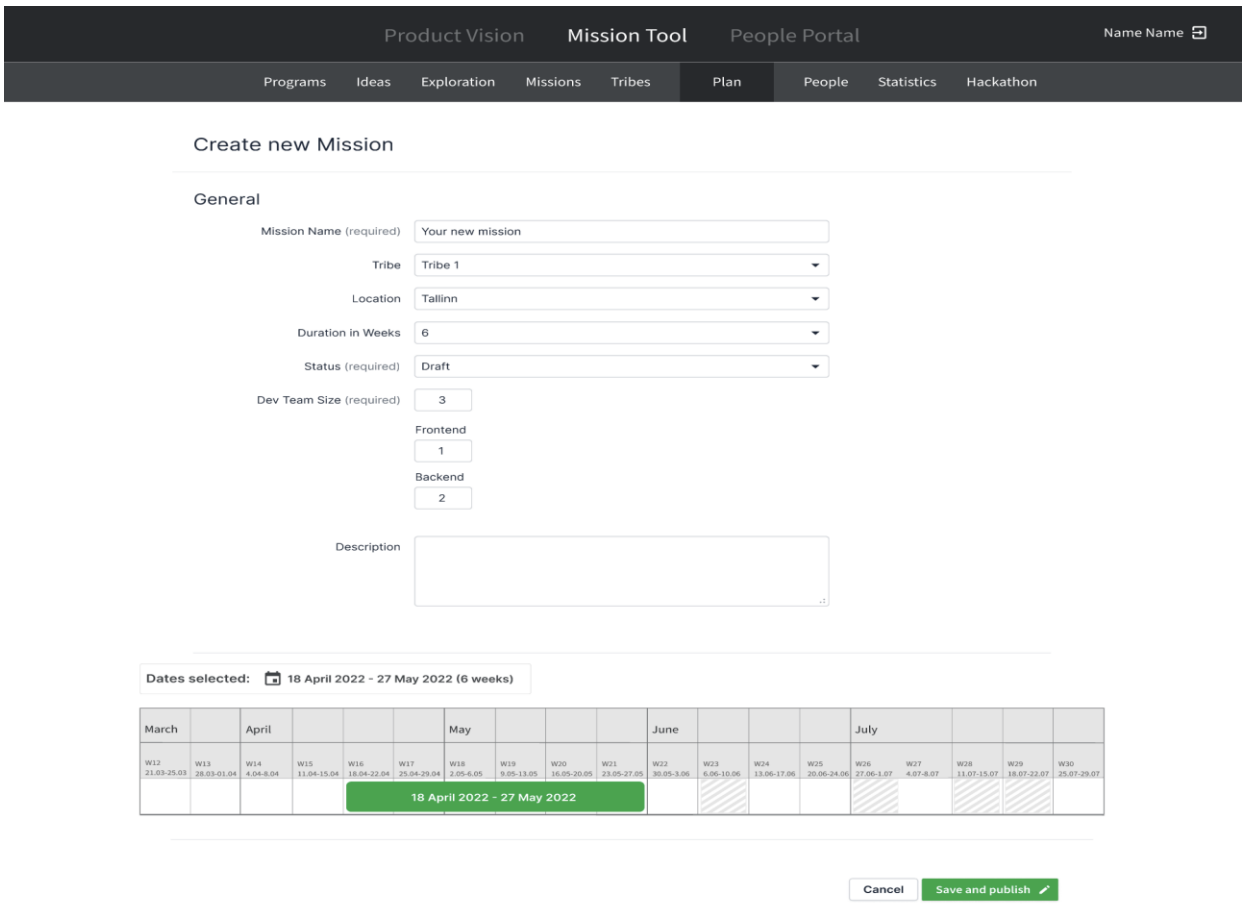
	hiirega soovitud projektile ja klõpsab sellel, saades selle tulemusel detailset infot. URL: "planning/tribe/:id"

Tabel 2. Projekti detailse sisu vaatamine

3.6.3 Uue projekti lisamine

Nimetus	Uue projekti lisamine
Eesmärk	Kasutaja saab lisada uusi projekte. Projekti lisamisel on vaja määrata projekti nimi, arendusüksuse nimetus, asukoha info, projekti kestus nädalates, staatus, arendusmeeskonna suurus, sh kui palju on meeskonda vaja esirakenduse arendajaid ning kui palju tagarakenduse arendajaid, ning kirjeldus. Samuti saab määrata kuupäevade vahemiku, millal soovitakse projekti läbi viia.
Tegevuse kirjeldus	Kasutaja navigeerib *“Planning”* nime all olevale kindla meeskonna planeerimislehele, kus on väljas kogu valitud meeskonnaga seotud statistika ning valitud meeskonnas toimuvad missioonid. Seal klõpsab ta rohelisel *“+”* nupul, mille tulemusena avaneb uus leht, millele ilmub vorm kõigi vajalike väljadega uue projekti loomiseks. Kasutaja täidab kõik nõutud väljad ja vajutab nuppu *“Save”*, mille tulemusena viib rakendus ta tagasi eelmisele statistikaga lehele, kus ilmus just kasutaja poolt lisatud projekt. URL: "planning/:tribeId/add"

Tabel 3. Uue projekti lisamine

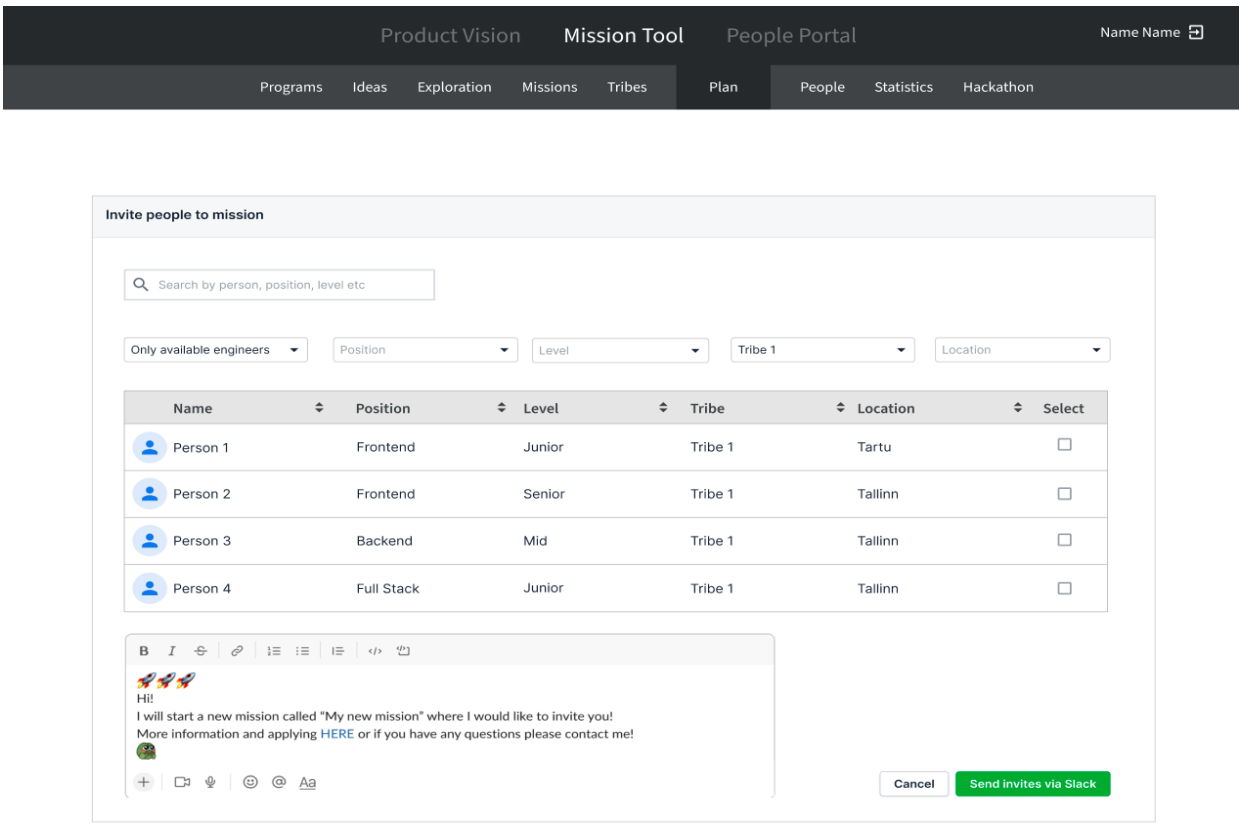


Joonis 3. Uue projekti lisamise vaade prototüüpist [7]

3.6.4 Inimeste kutsumine projekti

Nimetus	Inimeste kutsumine projekti
Eesmärk	Kasutaja saab kutsuda projekti inimesi, filtreerides inimesed saadavuse, positsiooni, taseme, arendusüksuse ja asukoha põhiselt ning vajadusel täiendades automaatselt genereeritud kutsumise sõnumit. Samuti on võimalik otsida inimest nime, positsiooni ja taseme järgi.
Tegevuse kirjeldus	Kasutaja navigeerib *"Planning"* nime all olevale kindla meeskonna planeerimislehele, kus on väljas kogu valitud meeskonnaga seotud statistika ning valitud meeskonnas toimuvad missiooni. Seal valib ta soovitud projekti ja viib hiirega selle peale, mille tulemusena ilmub projekti uute inimeste lisamise ikoon, millele klõpsates suunatakse kasutaja inimeste kutsumise lehele. Peale sobivate kandidaatide valimist sisestab kasutaja sõnumi, mis saadetakse kõikidele valitud kandidaatidele ning klõpsab nupul *"Invite"*, mille tulemusena saadetakse kõigile valitud kandidaatidele teade projektiga liitumise kutsega. Sõnum saadetakse kandidaadile isiklikult Slack'i suhtluskeskkonnas. Valida võib ühe või mitme kandidaadi vahel. URL: "missions/missionId/invite-people-to-mission"

Tabel 4. Inimeste kutsumine projekti



Joonis 4. Inimeste kutsumine projekti vaade prototüüpist [7]

4 Töö tulemus

Järgnevas peatükis tuuakse välja töö tulemus, milleks hetkel on täiendus juba olemasolevale Missions Tool'ile. Töö selle projekti kallal aga jätkub, kuna lõpptulemust pole veel saavutatud.

Hetkel on väljatöötatud lahenduse kasutajal võimalus hankida kogu vajalikku valitud meeskonnaga seonduvat infot, mis sisaldab iganädalast statistikat selle meeskonna töötajate hõivatuse kohta ning töötajate staatust käimasolevates projektides. Samuti saab kasutaja selle tööriista kaudu luua uue projekti, täites selleks välja töötatud lehel kogu vajaliku info, samuti saata ettevõttes kasutatavasse suhtluskeskkonda Slack sõnumi projektiga liitumisettepanekuga nii ühele kandidaadile kui ka mitmele.

4.1 Teostatud funktsionaalsused

Väljatöötatud lahendus on osa juba olemasolevast Mission Toolist ning kasutab eesmärkide saavutamiseks ka valmislahendusi.

Väljatöötatud lahendusel on neli põhifunktsiooni:

1. Annab kogu vajaliku statistika tulevase projekti planeerimiseks;
2. Annab teavet selle kohta, millised ressursid on juba käimasolevates ja kavandatavates projektides kaasatud;
3. Annab võimaluse luua uus projekt;
4. Annab võimaluse kutsuda projekti töötajaid.

Kõiki ülaltoodud funktsioone kirjeldatakse üksikasjalikult selle peatüki järgmistes osades.

4.1.1 Statistika ja projektide käimasolevate ressursid

Kogu iga projektiga seotud statistika ja ressurssidega seotud teabe kuvamiseks võeti aluseks lahendus, mille ettevõte töötas välja varem ühe teise projekti raames, mis ei kuulu käesoleva töö raamesse.

See lahendus on kalender, mille sisse on paigutatud lõppkasutajale vajalikud andmed. Kasutatav kalender on staatiline ja sellele lehele minnes kuvatakse selles olevad andmed. Lahendus töötab nii eelmiselt lehelt (kõikide käskude loendist) sellele lehele navigeerimisel kui ka URL'i abil otse

sellele lehele minnes. Mõlemal juhul küsib leht andmebaasist andmeid kahe API päringu kaudu, saades vastuseks kõik vajalikud andmed kulunud aja statistika projekteerimiseks, samuti andmed taotletud meeskonnas käimasolevate projektide kohta. Tulevase perioodi andmed arvutatakse kalendris teabe kuvamiseks vajaliku andmevormingu moodustamisel.

Kogu ülalmainitud API päringute käigus saadud info vajab ümberstruktureerimist, muidu ei saa kalender aru, kuhu, kuidas ja mida projekteerida. Selle loogika – andmete ümbervormindamise loogika – arendus töötati välja selle töö raames, mida selles osas kirjeldatakse.

Kõik statistikakalendris olevad andmed on esitatud järgmiste rubriikide all: Inseneride koguarv (*Engineers Headcount*), olemasolevate lahenduste juhtimiseks vajalike inseneride minimaalne arv (*Launchpad Minimum Size*), olemasolevate lahenduste kontrollimiseks vajalike inseneride arv (*Launchpad Actual Size*), juba kavandatud (*Engineers On Mission*) ja käimasolevate projektide jaoks vajalik inseneride arv (*Need for Missions*), vabade töökohtade arv projektides (*Open Slots on Mission*), valitud meeskonda mittekuuluvate inseneride arv (*Outtribers On Mission*), puhkusel olevate inseneride arv (*Vacation*) ja projektidega töötamiseks vabade inseneride arv (*Available for Missions*).

Mõned andmed, mis saadakse API päringu tulemusel, vajavad täiendavaid arvutusi.

Arvutused:

- Inseneride koguarv – ei ole vaja ümber arvutada
- Praeguse lahenduste juhtimiseks vajalike inseneride minimaalne arv – ei ole vaja ümber arvutada
- Olemasolevate lahenduste kontrollimiseks vajalike inseneride arv – ei ole vaja ümber arvutada
- Kavandatud inseneride arv – võetakse kõigi valitud meeskonna projektides korraga osalenud inseneride summa
- Käimasolevate projektide jaoks vajalike inseneride arv – võtab valitud meeskonna projektides korraga arendamiseks vajalike inseneride arvu
- Vabade töökohtade arv projektides – arenduseks vajalike inseneride arvust lahutatakse kõigi valitud meeskonna projektides korraga osalenud inseneride summa

- Puhkusel olevate inseneride arv – võetakse puhkusel viibivate töötajate arv sellel nädalal, mille eest puhkuste arv on arvestatud.

- Projektidega töötamiseks vabade inseneride arv – meeskonna töötajate arvust lahutatakse olemasolevate lahenduste kontrollimiseks vajalike inseneride arv, kavandatud inseneride arv ja puhkusel olevate inseneride arv.

Kõik väärtused arvutatakse iga nädala kohta. Kogu arvestusperiood on kolm aastat.

Andmebaasist statistika saamiseks kasutati juba olemasolevat API päringut, kuid seda täiendati, kuna see andis andmeid iga päeva kohta, eesmärgi saavutamiseks tuleb aga andmed esitada iga nädala kohta. Et kasutajaliidest selle ülesandega mitte üle koormata, viidi soovitud andmeid koguva ja tagastava funktsiooni sees läbi ümberkujundus, et koondada andmed nädalagruppidesse ja tagastada need soovitud vormingus.

Andmete kalendrile sobivasse vormingusse vormindamise meetod võeti kasutusele nullist, kuna varem kalendrisse projitseeritud andmed ei näe välja sellised, nagu on projekteeritud rakendatud lahenduses. Selle loogika põhiülesanne on kasutajaliidese poolt probleemideta loetavate andmete arvutamine, levitamine, järjestamine ja tagastamine, mille tulemusena jaotatakse kogu statistika edukalt üle kalendri. Selleks, et kalender paigutaks andmed õigesse kohta, peab igal andmetel olema alguspunkt – parameeter „*left*“ ja lõpp-punkt – parameeter „*width*“. Samuti peab kalender aru saama, millisel real peaksid andmed asuma – see polegi nii keeruline, sest igal real on laius, mis on järgmise rea alguspunktiks jne.

Kõik ülaltoodud andmed arvutatakse, genereeritakse ja tagastatakse ühes loogilises ahelas.

STATISTICS	March			April				May				June					July	
	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W22	W23	W24	W25	W26	W27	W28
Engineers Headcount	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
Launchpad Minimum Size	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Launchpad Actual Size	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
Need for Missions	3	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Engineers on Mission	5	5	5	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0
Open Slots on Mission	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Outtribers on Mission	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Vacation	1	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Available for Missions	8	8	7	11	10	14	14	14	14	14	14	14	14	14	14	14	14	14
MISSIONS (3)																		
Project 1	3/?	3/?	3/?	3/?	3/?	3/?	3/?	3/?	3/?									
Projekt 2	2/3	2/3	2/3															
Projekt 3																		

Joonis 5. Realiseeritud statistika vaade [15]

4.1.2 Uue projekti loomine

Seda funktsiooni Mission Toolis varem ei rakendatud. Lõpptulemus oli täiesti nullist välja töötatud ja mõjutab nii serveri poolt kui ka kasutajaliidest.

See funktsioon on POST API päring, mis loob andmebaasi tabelisse kirje uue projekti kohta ja kui see on edukalt loodud, tagastab kinnituse, et kirje loodi.

Kohe alguses lisati ruuterile uus API lõpp-punkt, kus kontrolliti sissetuleva päringu saatnud kasutaja volituse kehtivust ning edastati andmed kontrollerile.

Kontrolleris kontrollitakse sissetulevad andmed ja saadetakse seejärel meetodile, et luua andmebaasis uus kirje. Sõltuvalt andmebaasis uue kirje loomise tulemusest tagastatakse kas kinnitus uue kirje loomise kohta või uue kirje loomise ebaõnnestumise põhjus. See vastus läheb kontrolleri kaudu tagasi ruuterisse ja ruuterist kasutajaliidesesse.

Ülaltoodud rakendamine oli esimene samm selle funktsiooni väljatöötamisel. Järgmise sammuna töötati välja süsteem, mis saadaks ja võtaks vastu taotlusi kasutajaliidese poolel. Seda arutatakse edasi.

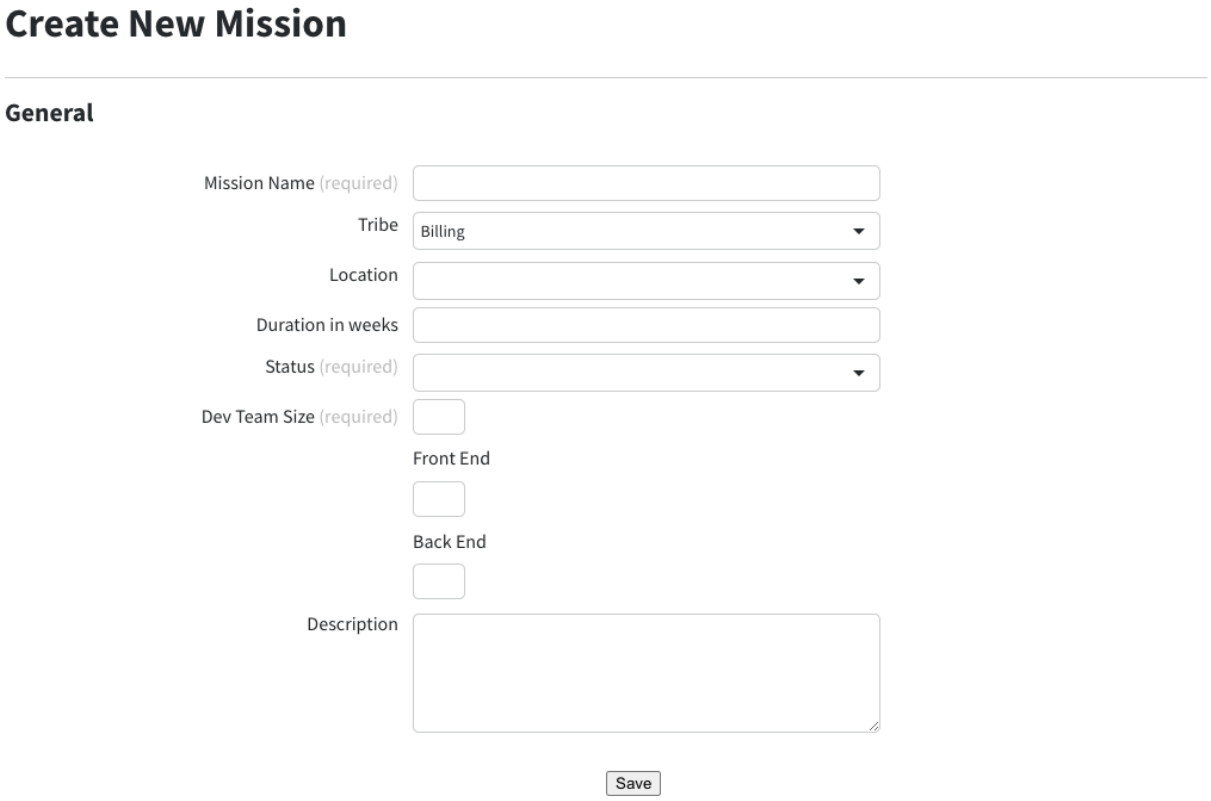
Kuna Missions Tool kasutab valdavalt ainult API GET-meetodid, siis uue projekti loomise ja lisamise funktsionaalsuse arendamise teine etapp loodi samuti nullist. See samm mõjutab kasutajaliidest, kuid töö tehti ainult selle liidese ja serveri vahelise suhtluse meetodil.

Kõik kasutajaliidese meetodid päringu serverisse saatmiseks on realiseeritud laienduse nimega Axios, mida seda tüüpi rakendustes laialdaselt kasutatakse. Kõik sai alguse uue *„action'i“* loomisest, mis võtab andmed kasutajaliidest ja edastab need meetodile, mis asub kõigi kasutajaliidese taotluste API kogus, et saata andmed serverisse. Kuna andmete valideerimine toimub nii enne andmete „*action*“ edastamist kui ka serveris endas, siis selles väljatöötatud lahenduse osas andmete valideerimist ei nõutud.

Kuna eelpool oli märgitud, et kõik selles ülesandes töötati välja nullist, siis tuli ka see meetod juurutada, mis sai tehtud. Meetodi funktsioon on päringute saatmine serverisse, mis tuli täita päringu saatmise lõpp-punkti, saadetud päringu tüübi, mis on POST-meetod, juurdepääsuvõtme ja andmete endaga – see kõik on juba olemas meetodis, kui seda kutsutakse, välja arvatud andmed, mis edastatakse rakendusest *„action'i“*. Pärast seda saabuvad andmed kirjeldatud osasse enne seda.

Kuna see oli kogu uue projekti loomiseks vajalik toimingute ahel, siis viimaseks sammuks oli ülaltoodud funktsionaalsuse ühendamine uue projekti loomise lehel oleva kasutajaliidese salvestamisnupuga. Seda rakendati ülaltoodud komponendi „*action*“ valideerimisega ja sidudes selle nupuga „*save*“. Funktsionaalsus kutsutakse välja nupu klõpsamisel.

Kõik ülaltoodud lehel olevad komponendid ja ka leht ise töötati välja “Ettevõttesisese tarkvara kasutajaliidese täiendamine” töö raames.



Joonis 6. Realiseeritud uue projekti loomise vaade [15]

4.1.3 Inimeste kutsumine projekti

Seda funktsiooni Missions Tool’is varem ei rakendatud. Lõpptulemus oli täiesti nullist välja töötatud ja mõjutab nii serveri poolt kui ka kasutajaliidest.

See funktsioon on API POST päring, mis saadab inseneridele kutsed uue projekti arendusega liitumiseks ning loob andmebaasi ka kirje selle kohta, keda kutsuti.

Kuna see ülesanne osutus üsna mahukaks, otsustati see jagada neljaks osaks:

1. Rakendada Slack’i suhtluskeskkonnas töötajatele sõnumite saatmise teenus

2. Lisada uus tabel andmebaasi

3. Töötada välja uus API lõpp-punkt, et salvestada andmeid töötajatele saadetud kutsete kohta

4. Linkida ülaltoodud funktsioonid, kuna need peaksid koos töötama.

Eelnimetatud teenuse arendus sai alguse uue rakenduse loomisest suhtluskeskkonda Slack'i sees. Selleks oli vaja saata vastav päring läbi Slack'i rakenduste haldamise eriteenuse. Kuna see töökeskkond on mõeldud ettevõttesiseseks suhtlemiseks, peab kõik, mis sellesse keskkonda lisatakse, muudetud või sealt eemaldatud, olema kinnitatud kõigi ettevõttes kasutatavate tööriistade ja lahenduste väljatöötamisse ja hooldamisse kaasatud meeskonna poolt. Pärast rakenduse lisamise taotluse kinnitamist konfigureeriti see rakendus nii, et selle kasutamine oleks võimalik, ning sellel oli ka luba kõigi vajalike funktsioonide jaoks:

1. Võimalus saada ettevõttesisest suhtluskeskkonda kasutavate kasutajate nimekirja

2. Võimalus saada valitud kasutaja meili ettevõttesiseses suhtluskeskkonnas

3. Võimalus saata ettevõtte suhtluskeskkonnas valitud kasutajale rakenduse nimel isiklik sõnum.

Kuna ülaltoodud funktsioonid on lihtsalt teostatavad, pole nende kasutamiseks lisaluba vaja. Lisatud on ka selle rakenduse kirjeldus, selle ülesanne, nimi ja muud andmed.

Pärast kõiki seadistusi genereeriti pääsuvõti loodud rakenduse nimel ettevõtte suhtluskeskkonnas toimingute tegemiseks.

Järgmise sammuna juurutati ettevõtte suhtluskeskkonda sõnumite saatmise teenus, mis sai alguse uue teenuse deklareerimisest veebikliendiks arendatud rakenduses. Selleks lisati uus teenuse loomise deklaratsiooni funktsioon, lisati see funktsioon teenuste loomiseks kutsutavate funktsioonide loendisse ja deklareeriti kõik vajalikud muutujad spetsiaalselt selleks loodud failis. Tulemuseks oli töötav teenus, mis sai alguse kogu rakenduse käivitamise ajal.

Järgmiseks oli vaja realiseerida funktsioonid, mis saadaksid kasutajale sõnumi, samuti funktsioon, mis otsiks kõigi sidemeediumi kasutavate kasutajate loendist kasutajatunnust. Kasutajatunnuse otsingu funktsiooni loomine on vajalik selleks, et rakendus saaks aru, kes peab sõnumi saatma, seda ID-d on vaja ka sõnumi saatmise päringu API skeemis ning selle põhjuseks oli asjaolu, et ID andmebaasis ja ID suhtluskeskkonnas on erinevad. Otsinguelement oli selle kasutaja meiliaadress, kellele soovitakse sõnumi saata. See meiliaadress pärineb funktsioonist, mida kirjeldatakse järgnevalt. Mõlemad funktsioonid on API GET päringud, millele vastuseks saab

suhtluskeskkonnas kasutaja ID hankimise funktsioon ID, kui läbitud meiliga kasutaja on olemas, samas kasutajale privaatsõnumi saatmise funktsioon arendatud rakenduse nimel tagastab loodud sõnumi oleku.

Kahe ülaltoodud funktsiooni ühendamiseks realiseeriti veel üks funktsioon, mis andis välja töötatud lahenduse serveriosa kontrolleris väljakutse nendele funktsioonidele korraga. See meetod sai sisendiks kolm funktsiooni, mida kutsutakse selle probleemi lahendamiseks vajalikus järjekorras:

1. Andmebaasis üles otsimine projekti kutsutud töötaja meiliaadress;
2. Funktsiooni kutsumine ettevõtte suhtluskeskkonnas kutsutud isiku ID otsimiseks
3. Pärast eelmises etapis kutsutu ID edukat otsimist saata sellele inimesele loodud rakenduse nimel isiklik sõnum kutsega
4. Tagastada lõpetatud toimingute oleku.

Esimene samm ülaltoodud protseduuris on vajalik, kuna ID andmebaasis ja ID suhtluskeskkonnas on erinevad, kuid mõlemas kohas kasutatav e-mail on sama, mis võimaldas antud lahendust realiseerida.

Niipea kui ülaltoodud funktsioonid rakendati, jäi üle vaid lisada ruuterisse uus POST-päring ja sellele vajalik kontroller. Need toimingud on identsed punktis 4.1.3 rakendatutega, kuid sellel teostusel on järgnev omapära. Kui punktis 4.1.3 oli kontrolleril üks ülesanne, siis selles osas väljatöötatud kontrolleril on neid kaks:

1. esimene osa on kasutajale kutsete saatmine ettevõtte suhtluskeskkonnas
2. kasutajale kutse saatmise kohta andmebaasi kirje lisamine, millest tuleb juttu järgmisena.

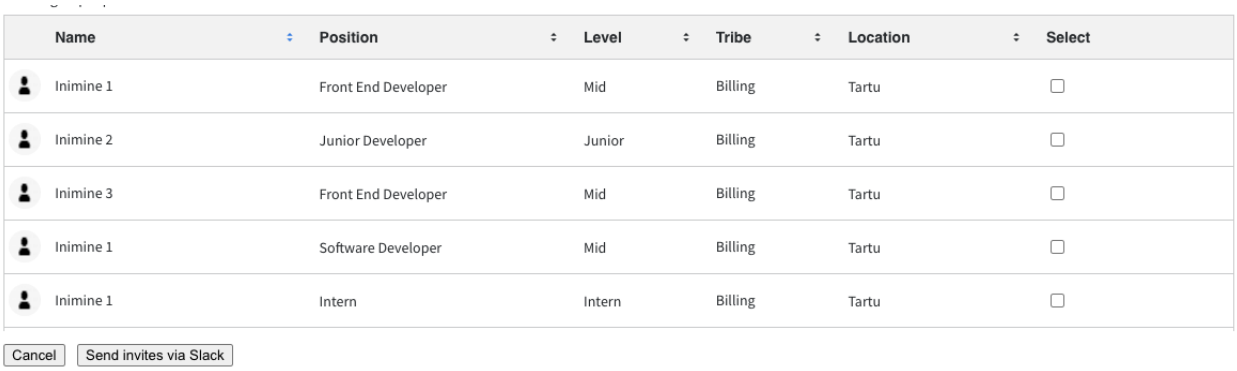
Kuna varem ei olnud võimalik automaatselt kutsuda inimesi projekti osalema, siis seda funktsiooni ka ei loodud. Seda silmas pidades loodi saadetud kutsete kohta info salvestamise süsteemi väljatöötamise alguses rakenduse poolt kasutatavasse andmebaasi uus tabel. Selle loomiseks lisati uus migratsioon kogu vajaliku teabega. Migratsiooni rakendamiseks ja selle mõistmiseks MySQL struktuuri järgi välja töötatud andmebaasi abil kasutati funktsionaalset raamatukogu nimega Knex. Sellest raamatukogust oli 3.3. osas varem juttu.

Tabeli loomisel jäi üle rakendada meetodit sellesse tabelisse uue kirje lisamiseks. See meetod töötati välja sama teeki abil, mida kasutati tabeli loomisel – Knex. Meetod lisab iga toimingu kohta ühe kirje. Kuna eelnevalt sai mainitud, et kutsuda saab korraga nii ühte osalejat kui ka mitut, siis

selle funktsiooni käigus tehtud otsus lähtus sellest, et infot on vaja saada iga lisamise kohta eraldi, mitte korraga, kuna kasutaja, kes kutsub projekti osalejaid, peab saama tagasisidet selle kohta keda lisati ja keda mitte.

Ja nüüd, kui kõik vajalikud komponendid projekti osalejate kutsumiseks olid valmis, jäi üle vaid kaks ülaltoodud funktsionaalsust üheks ühendada. Seda tehti kontrolleris, mis loodi varem Slack'i suhtluskeskkonnas kandidaadile sõnumi saatmise teenuse arendamise käigus. Kontrolleris proovib see esmalt saata kutse Slack'i suhtluskeskkonda ja kui see õnnestub, siis luuakse selle kohta kanne selle tööriista poolt kasutatavasse andmebaasi tabelisse. Selline järjekord valiti seetõttu, et andmebaasis oleva kirje eesmärk on salvestada teavet selle kohta, kellele kutse täpselt saadeti, mitte selle kohta, kellele seda ainult saata prooviti. Nagu varem mainitud, on tänu *FOR-tsükli* kontrollerisse integreerimisele võimalik saata kutseid mitmele kandidaadile korraga.

Sellega viidi lõpule kandidaatide projekti kutsumise teenuse arendamine. Selle töö teostamise ajal kasutajaliidesega suhtlemist ei toimunud, kuna valmisoleku hetkel ei olnud see veel täielikult valmis, kuna selle töö tegemiseks oli aega piiratud. Selle süsteemi arendamise protsess kasutajaliidese raames on leitav *"Ettevõttesisese tarkvara kasutajaliidese täiendamine"* tööst.



Joonis 7. Realiseeritud inimesi kutsumise vaade [15]

4.2 Koodi arhitektuur ja mustrid

Kuna antud töö ülesandeks oli juurutada kõik rakenduse serveriosas töötamiseks vajalikud komponendid, samuti välja töötada mõned funktsioonid kasutajaliidesele vajaliku info edastamiseks, kirjeldatakse selles osas vaid üldist infot ja sellega seonduvat. Seatud eesmärkidele. Teave kasutajaliidese arhitektuuri kohta esitatakse *"Ettevõttesisese tarkvara kasutajaliidese täiendamine"* töö osana.

4.2.1 Arhitektuur

Kogu arendatud funktsionaalsus ja ka rakendus ise on salvestatud eraldi *mission-tracking* hoidlasse, mis asub Github'i keskkonnas. Rakendus koosneb kahest peamisest projektifailist – taustaprogrammist ja kasutajaliidest, samas kui testid on eraldi fail. Kogu serveriosa on välja töötatud enne käesolevas töös kirjeldatud projekti elluviimist ja ka kasutajaliides, kuid neid kõiki täiendati eraldi olemasoleva funktsionaalsusega, mida võib käsitleda eraldi olemasolevate süsteemidena. Arendatud funktsionaalsus ei olnud enamasti varem kasutatud skeemi järgi üles ehitatud, kuid siiski on väikeseid osi, mis on siiski sarnased, kuna uute juurutamisel oli sarnane funktsionaalsus, kuid see oli seotud kas muude andmebaasi tabelitega või muude teenustega.

Samuti täiendati kogu väljatöötatud funktsionaalsust nii ühiku- kui ka funktsionaalsete testidega. Nende rakendamiseks loodi selleks sobivasse kohta kaustad nimedega *"Tests"* ja *"__Tests__"* – funktsionaalsete testide tuvastamiseks on vaja nime koos elementidega *"__"*. Selline otsus on rakendusesisese failistruktuuri alusel kohustuslik. 5. Peatükis käsitletakse teste üksikasjalikumalt.

Samuti tuleb failistruktuurile viidates märkida, et kogu failistruktuur on välja töötatud ja ka arendusprotsessis toetatud vastavalt *"Clean Code"* põhimõtetele – kõik failid on jagatud kaustadesse, millel on ainult teatud tüüpi või teatud failid, funktsionaalsus ja ka nimi , nii failid kui ka funktsioonid, muutujad ja muud kasutatud koodiosad nimetati kõiki *"Clean Code"* põhimõtte reegleid järgides.

Kõik väljatöötatud funktsionaalsused on kasutusel „*Planning*" rakenduses, mis sisaldub Mission Tool'i üldfunktsionaalsuses. Veebilehel „*Planning*" on kaks moodulit, kus kummaski on kalendrid – ühes moodulis näidatakse kalendrit koos iga nädalase statistikaga, teises moodulis aga käimasolevate või tulevaste missioonide oleku projitseerimiseks. Kõik lahenduses olevad andmed esitatakse valitud käsu alusel. Iga statistikakategooria ja selle arvutamise täielik kirjeldus on esitatud punktis 4.1.1. Samuti töötati välja töötatud funktsionaalsuse juurutamiseks *"Ettevõttesisese tarkvara kasutajaliidese täiendamine"* töö käigus välja komponendid, mille abil viidi läbi arendatud funktsionaalsuse kasutamine. Uue projekti lisamise protsessi alustamiseks tuleb kasutajaliidesest edastada meeskonna ID, projekti nimi, projekti olek, vajalike arendajate arv, projekti algus- ja lõppkuupäevad ning soovi korral saab lisada projekti kirjelduse ja asukoha. Projektivoogu arendaja kutse saatmise käivitamiseks tuleb kasutajaliidesest edastada kutsutu/kutsutud inimeste ID, kutsuja ID, projekti ID ja kutsutavale saadetav sõnum.

Serveri osa töötab vastavalt skeemile *ruuter-kontroller-andmebaas/muu teenus*.

4.2.2 Kood

Lahenduskood tulevaste projektide ajastamiseks on jagatud serveri osaks ja kasutajaliideseks. Lahenduse serveriosa töötati välja JavaScript programmeerimiskeele ja express.js raamistiku abil. Kasutati ka palju erinevaid abiraamatukogusid, mis olid ära toodud töö 3.3. osas. *"Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine"* töös kirjeldati, mille kasutamist kasutajaliides töötati välja.

Osa serverirakenduse koodist realiseeriti olemasolevate lahenduste abil ja teine töötati täielikult välja nullist. Olemasolevatele lahendustele ja nende kasutamisele viidates tähendab see enamasti seda, et lahendus on võetud eeskujuks konkreetse funktsionaalsuse juurutamisel, kuid mõnel juhul sai seda kasutada kujul, mis töötati välja enne selle projektiga töö alustamist ja täiustatud, et võimaldada seda kasutada ettenähtud otstarbel.

Taustaprogrammis on kokku 7 kausta:

- controllers – see kaust sisaldab kõiki teenuste toimimise, andmebaasiga suhtlemise ja sellega seotud toimingute eest vastutavaid kontrollereid;
- cron-jobs – see kaust sisaldab kõiki töid, mis teatud aja möödudes tehakse. Näiteks töö, mis on seotud kõigi töötajate teenuse olemasolevate töötajate sünkroonimisega;
- graphql – mõningaid teenuseid, mis ei ole selle projekti arendusse kaasatud, kasutatakse andmete pärimiseks andmebaasist vastavalt GraphQL skeemile. See kaust sisaldab meetodeid andmete pärimiseks vastavalt GraphQL-i skeemile;
- middleware – see kaust sisaldab Pipedrive'i inseneride poolt välja töötatud komponente, millest üks on näiteks metoodika info edastamiseks loggeris;
- routes – see kaust sisaldab kõiki API lõpp-punkte iga teenuse jaoks eraldi;
- services – see kaust sisaldab iga selles tööriistas kasutatava teenuse funktsioone;
- utils – see kaust sisaldab lisafunktsioone, mida näiteks kasutatakse mitmes komponendis.

Kõigil tehtud HTTP-päringutel on konkreetsed nimed, mis kajastavad nende käitumist ja eesmärki. Näiteks GET-päringu puhul, mis peaks tagastama loendi kõigist andmebaasis olevatest konkreetse meeskonna töötajatest, oleks URL järgmisel kujul: "api/tribes/id/people". Andmete saamiseks andmebaasist võetakse päringust käsu id ja kontrolleri abil kutsutakse välja vajalik SQL päring.

Kõik SQL-päringud salvestatakse andmebaasi kausta, mis sisaldab faili kõigi võimalike päringutega iga tabeli kohta. Kaust asub teenuste kaustas, kuna andmebaas on tegelikult ka teenus. Kõigi meeskonnaliikmete kohta teabe saamiseks ülaltoodud päringu kutsumisel kutsutakse näiteks meetod *getPeopleByTribe()*, mis omakorda moodustab SQL *"WHERE-klausli"* lisandi teises meetodis, mida nimetatakse *getPeopleBy()*-ks, mis on kirjeldatud päringu lõpp-punkt. Kõik selles taotluses mainitud meetodid on asünkroonsed ja töötavad asünkroonse funktsiooni reeglite kohaselt.

Töö käigus oli vaja ka andmebaasi uue tabeliga täiendada. Seda tehti uue migratsiooniga. Migratsioonide jaoks on loodud ka eraldi kaust, kuhu salvestatakse kõik kunagi tehtud migratsioonid. Iga migratsiooni rakendatakse abiteegi Knex abil, mida kirjeldati selles osas.

Uue tabeli lisamiseks koostati SQL-lause, mis konverteeriti koodiks abiteegi Knex abil. Pärast migratsiooni loomist peate lihtsalt serverirakenduse käivitama, süsteem on loodud selliselt, et see parandab uue migratsiooni ise.

4.2.3 Mustrid

Muster on läbiproovitud lahendus tüüp-probleemile ning on kasutatav erinevates kontekstides. Järgnevas peatükis analüüsime *C. Larmani* raamatu *"Applying UML and Patterns"* [17] ja *R. Harmes* raamatu *„Pro JavaScript™ Design Patterns"* [18] põhjal, kuidas projektis on rakendatud "GRASP" ja "Singleton" mustreid.

4.2.3.1 GRASP – Informatsiooni ekspert

Ekspert on klass, mis omab teatud vastutuse täitmiseks vajalikku informatsiooni. Iga klass on mingi vastutuse osas ekspert, näiteks Mission objekti loomiseks on vajalik *confluence.js* klassi olemasolu, seega Confluence klass on ekspert.

Informatsiooni eksperdiks on antud koodil *confluence.js* ja *bamboo.js,* mis saavad kätte ja edastavad infot API'de kaudu.

4.2.3.2 Singleton muster – Üks ja ainuke

See on oluline stsenaariumi korral, kus on vaja luua ainult üks eksemplar, näiteks andmebaasiühendus. Eksemplari saate luua ainult siis, kui ühendus on suletud, või sulgege kindlasti avatud eksemplar enne uue avamist. Seda mustrit nimetatakse ka rangeks mustriks. Üks

selle mustriga seotud puudusi on selle raske testimiskogemus, mis on tingitud peidetud sõltuvusobjektidest, mida pole testimiseks lihtne eraldada.

Üks ja ainuke lahendus arendatud projektis on lahendus, mis käivitab kõik arendatud projekti tööga seotud teenused, mis asub selles failis. Enne juba töötava sellise lahenduse peatamist ei saa uut käivitada.

4.3 Koodimeetrika

Pipedrives on kasutusel SonarQube, mis on koodi kvaliteedi tagamise tööriist. SonarQube analüüsib lähtekoodi ning esitab aruandeid projekti koodi kvaliteedi kohta. See ühendab endas staatilised ja dünaamilised analüüsitööriistad ning võimaldab kvaliteeti aja jooksul pidevalt mõõta. Iga kord kui kood commititakse läbib see SonarQube analüüsi, nii saab arendaja kohe tagasiside, kas koodis on duplikeerimist või buge ning hinnangu koodi hooldatavuse kohta. Samuti näitab SonarQube koodi kaetuse testidega ning annab teada mitu rida on veel vaja testida [19].

Lisaks SonarQube'le on Pipedrives kasutusel ka CodeShip, mis on pilvepõhine rakenduste arendus platvorm. Selle eesmärk on aidata ettevõtetel Node rakendusi integreerida ja juurutada. CodeShip võimaldab Pipedrive'il kohandada veebirakendusi ning automatiseerida platvorme, millel puuduvad serverid. Pipedrive arendajad valivad, millised sammud peaksid jooksma järjest või paralleelselt ning mitu tegevust peab samaaegselt käivitama [20].

Selle projekti arendamise käigus oli kirjutatud kokku ~1500 rida koodi, mis on jäänud alles peale koodi refaktoorimist. Kokku kirjutatud ligikaudu ~3000 rida koodi, mida oli pidevalt muutnud ning parandanud, et kood vastaks *"Clean Code"* põhimõtetele. Oli eemaldatud nii selle projekti arendamise käigus kirjutatud kui ka juba olemasolevast koodist duplikeerimist 500 rea jagu.

Sellele lisaks oli kirjutatud 50 funktsionaalset ja 18 üksiku testi. Selle projekti arendamise käigus kirjutatud koodist on 90% kaetud testidega, mis koosneb juba eelnevalt olemasolevatest testidest kui ka juurde kirjutatud testidest.

5 Testimine

Tulevaste projektide planeerimise lahenduse väljatöötamise käigus viidi ettevõtte sees läbi nii üksuse- kui ka funktsionaalsed testid. Testid katavad projekti elluviimise ajal 90% väljatöötatud koodist. Kaks eraldi testprojekti – üks üksuste testide jaoks, üks funktsionaalsete testide jaoks. Iga teenuse või komponendi jaoks on loodud eraldi testprojektid. Kõiki testprojektide kogusid käivitatakse koos – kõik ühikutestid koos ja kõik funktsionaalsed testid koos.

Ühiktestide kirjutamiseks kasutati Jest raamistikku ja funktsionaalseteks testideks Chai't [21], [22]. Eelnevalt mainitud raamistike arhitektuur on üsna sarnane, kuid üht kasutatakse sagedamini ühiktestidega ja teist funktsionaalsete ning nende raamistike valikut mõjutas asjaolu, et neid raamistikke kasutati ettevõttesiseselt.

Iga testide kogum oli funktsioonis *„describe“*, mis sisaldas palju *„it“* funktsioonide teste. Mis tahes andmekogu põhjal testimisel kasutati *“for”*- või *“forEach”-tsükleid*.

Testid kirjutati AAA mustri järgi.

5.1 Ühiktestid

Ühiktestid on mõeldud väikese koodijupi testimiseks ja selle kontrollimiseks, kas see täidab ettenähtud funktsiooni õigesti. Käesolevas töös välja töötatud ühiktestid olid ainult planeeringulahenduses kasutatud kalendri statistikas toodud meetodile. Testiti kõiki ülaltoodud meetodiga seotud funktsioone – 12 eraldi meetodit, mis on omavahel seotud.

Ühikutestide jaoks on loodud eraldi sisendid, mis ei ole andmebaasiga seotud, kuna see on vaid meetod sisendi arvutamiseks ja ümbervormindamiseks. Järgmisena käsitlakse testi, mis vastutab õigete andmete tagastamise kontrollimise eest nende arvutamise ajal.

Arvutuste tulemusel tagastatud andmete kontrollimise testi rakendatakse kõikidele kalendris võimalikele andmeüksustele. Sellele viitab enne testi lisatud funktsioon *“each”*, milles on sisendandmeteks samad võimaliku statistika rubriigid. Testi alguses deklareeritakse päring, mis koosneb kõigist õigete andmete arvutamiseks vajalikest parameetritest, mis seejärel saadetakse statistika arvutamise meetodi sisendina, saades väljundina vajaliku väärtuse. Arvutamise käigus saadud väärtust võrreldakse iga katsetulemuse kontrollimiseks eelnevalt deklareeritud õigete väärtustega. Võrdlus toimub *"expect"* ja *"toBe"* funktsioonide kombineerimise teel:

- expect – sisendiks on testi käigus saadud väärtus
- toBe – sisend on väärtus, mis peaks olema testi käigus saadud väärtus.

Ülaltoodud testi näite leiatav lisast 2.

Kõik arendatud lahenduses olevad ühikutestid on üsna sarnased ülaltoodud näidistestiga, et arvutuste tulemusel tagastatud andmete kontrollimist rakendatakse kõikidele võimalikele kalendri andmeüksustele. Testid on realiseeritud selleks, et kontrollida vastuvõetud andmeid, kontrollida funktsiooni õiget käitumist ebaõigete sisendandmete saamise korral, kontrollida, kas sisendandmed on tühjad, üks või mitu jne. Enamasti on testis ainult üks *"expect"*, mõnikord kasutatakse aga rohkem kui ühte, näiteks testis, mis kontrollib tagastatava andmevormingu õigsust ja objekti struktuuri, milles andmed tagastatakse.

5.2 Funktsionaalsed testid

Funktsionaalsed testid on mõeldud kogu süsteemi või teenuse tervise testimiseks. Testimise ajal tuleks kontrollida kõiki võimalikke tulemusi: nii positiivseid kui ka negatiivseid. Selle töö käigus välja töötatud funktsionaalsed testid on rakendatud nii, et neid saab kasutada nii ühe kui ka teise teenuse jaoks. See oli võimalik tänu sellele, et mõlemas teenuses tuli testida ainult POST-meetodeid. Testiti kõiki väljatöötatud teenuste võimalikke kasutusjuhtumeid, kuid ainult neid teenuste osi, mis on andmebaasiga ühendatud, kuna teised osad kasutavad kolmanda osapoole teenuseid, mida kohapeal testida ei saa.

Funktsionaalsete testide jaoks luuakse nende käivitamisel lokaalne andmebaas, kuhu lisatakse kõik vajalikud tabelid spetsiaalselt kujundatud andmetega, mida kasutatakse konkreetse teenuse testimiseks. Andmebaasi tabelites olevatel andmetel pole midagi pistmist tegeliku andmebaasi tegelike andmetega. Järgmisena käsitletakse testi, mis kontrollib, et teenused ei saaks luua andmebaasis uut kirjet ilma kõigi nõutavate muutujateta.

Kuna funktsionaalsete testide väljatöötamisel kasutataval raamistikul pole sellist funktsiooni nagu *"each"*, kasutati standardset tsüklilist funktsiooni *"forEach"*. Seda tehti selleks, et rakendada ülalkirjeldatud väljatöötatud testi mõlemale arendatud teenusele. Selle meetodi valimisel välistati koodi dubleerimine ning realiseeriti võimalus kõik vajalikud seadistused ja andmed lihtsalt testile edastada. Testi põhiolemus on saada saadetud päringule vastus, mis sisaldab teadet uue kirje loomise ebaõnnestumise kohta andmebaasi tabelis ja sisaldab ka HTTP-olekukoodi 400. See käib nii: test saab päringu saatmiseks vajalikud valikud, misjärel ootab vastust ja rakendab sellele kaks

“await” funktsiooni – üks funktsioon kontrollib HTTP-oleku 400 olemasolu ja teine veateate olemasolu, kuna samuti kõik vajalikud andmed selle sees. Päringu valikud moodustuvad kogu testide kogumi käivitamisel, seda tehakse eelpool mainitud sissetulevate andmete põhjal. Samuti on kõik kollektsioonis olevad testid asünkroonsed, kuna tööd tehakse hunniku erinevate teenustega.

Samuti kustutatakse iga jooksva testi järel testi käigus loodud kanded andmebaasis.

Ülaltoodud testi näite leitav lisast 3.

Ülaltoodud näites on ainult kirjeldatud testi jaoks vajalikud andmed ja funktsioonid. Kõik selle töö käigus välja töötatud funktsionaalsed testid erinevad üksteisest ja neil on erinevad viisid korrektsuse kontrollimiseks. Peaaegu kõigil testidel on mitu *„expect“* elementi.

6 Analüüs ja järeldused

Antud osas tuleb analüüs tehtud tööle, meeskonnatööle, juhendamisele, ülevaade kasutatud kirjandusest ning kokkuvõtlikud järeldused.

6.1 Tehnilise teostuse analüüs

Rakenduse üldstruktuur ja arhitektuur lepiti kokku ettevõttes oleva projektijuhi, Mission Tool tööriista omaniku ja meeskonnale antud mentoritega, kes määrasid esmalt tulevase lahenduse tehnoloogiad, raamistikud, aga ka selle struktuuri. Iganädalastel koosolekutel kohandati tehnilist teostusplaani vastavalt seda projekti arendava meeskonna muutuvatele nõuetele ja oskustele.

6.1.1 Nõuded

Nõuded sellele lahendusele töötati välja ja kehtestati selle töö käigus. Nõuti, et kasutajale tehtud rakendus pole mitte ainult kasulik, vaid ka lihtne kasutada. Mugavuse huvides otsustati väljatöötatud lahendus võimalikult palju automatiseerida, säästa kasutajat nii palju kui võimalik käsitsitööst ning pakkuda lahenduses ka laia valikut funktsionaalsust. Kavandatav lahendus peaks olema võimalikult informatiivne, andes teavet üldisemalt ja üksikasjalikumalt. Kuna antud lahendus on osa suuremast süsteemist, mida edasi arendatakse, oli vaja seda arendada kasutajasõbralikult, aga ka tulevastele arendajatele võimalikult arusaadavalt.

6.1.2 Arhitektuur

Meeskonnaprojekti raames projekti väljatöötamisega seotud meeskond praktiliselt ei mõjutanud selle töö käigus olnud arendusobjekte, mistõttu pärast meeskonnaprojekti edukat kaitsmist asus meeskond selle jätku arendama, mida otsustati mõjutada ka serveri osa, tänu millele kogu see töö sündis. Projekti käigus puudutati peaaegu kõiki võimalikke võimalusi serveriosaga töötamiseks ning selle funktsionaalsuse võimalikke teostusi.

Kuna serveri osa meeskonnaprojekti käigus ei mõjutanud, nagu varem mainitud, siis võttis meeskonnal aega nii selle uurimine kui ka materjaliga tutvumine, mis andis arusaama, kuidas serveriga töötada ja mida selleks vaja on. . Selle aja andis asjaolu, et *“Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine”* töö raames viidi läbi prototüübi analüüs ja arendus. Analüüsi tulemusena otsustati teha andmebaasis mõned muudatused, uue teenuse loomine, aga ka olemasolevate täiendamine ja kaasajastamine, millest tuleb juttu edasi.

6.1.2.1 Andmebaas

Töö käigus kasutati lokaalse MySQL andmebaasi. Analüüsi tulemusena tehti otsus lisada uus tabel, mis on vajalik uue teenuse toimimiseks, mida kirjeldame hiljem. Analüüsi läbi viinud meeskond esitas uue tabeli jaoks kõik kriteeriumid ja esitas ka SQL'i ettepaneku selle loomiseks. Tabeli lisamiseks andmebaasi loodi ja lisati uus migratsioon, mis realiseeriti Knex'i teegil põhineva SQL-lausete koodiks tõlkimise meetodi järgi.

Sequel Ace'i kasutati kohaliku andmebaasi muudatuste juhtimiseks, hooldamiseks ja kinnitamiseks.

6.1.2.2 Teenused ja funktsioonid

Analüüsi käigus otsustati luua uus teenus, uuendada olemasolevat, samuti juurutada kalendrisse statistika esitamise loogika. Töö käigus selgus, et Slack'i töötajate suhtluskeskkonnas rakendatakse täiesti uudset projektiga liitumise kutse saatmise teenust. Uute funktsioonidega täiendati ka ettevõttes käimasolevate projektide andmete esitamise teenust, mis oli juba töö alustamise hetkel olemas. Loetletud teenustega töötamise käigus mõjutati kõiki nende tööga seotud mooduleid. Teenuse ruuteris rakendati kõik võimalikud päringu vastuvõtmise teed, kontrolleris loodi uued kontaktpunktid ja rakendati kõik võimalikud sissetulevate andmete valideerimise viisid. Tööd tehti ka nii olemasolevate päringutega andmebaasi kui ka täiesti uute lisamisega. Edukalt on välja töötatud ka kalendrisse statistika esitamise meetod.

Töökeskkonnaks ülalnimetatud teenustega töötamise käigus oli Microsofti programm Visual Studio Code.

6.1.3 Kood

Selle projekti elluviimisel kasutati paindlikke metoodikaid funktsionaalsuse ja süsteemide arendamiseks. Kogu kirjutatud kood rakendati *„Clean Code"* põhimõtete järgi. Kõik projektis juurutatud lahendused on kooskõlas üldise Mission Tooli kodeerimisstiiliga, samuti puuduvad piirangud kasutatava lahenduse juurutamise metoodika raames – kogu juurutatud materjal on taaskasutatav ja sellest pole raske aru saada. Eriti rasketel hetkedel jäeti kommentaarid nende hetkede täpsemaks mõistmiseks.

Kasutati funktsioonide ja muutujate nimetamise meetodeid, mis sisalduvad ka *"Clean Code"* põhimõtetes. Need põhimõtted on hästi kirjeldatud *M. Fowleri* raamatus *„Clean Code"*. Kõik väljatöötatud projekti nimed on konstruktiivsed ja annavad täpse ülevaate sellest, kus, miks ja

kuidas neid kasutati. Samuti kasutati selle lahenduse väljatöötamisel funktsioonide ja muutujate nimetamise reegleid *"lowerCamelCase"*, kuna see on JavaScript'i programmeerimiskeele kasutamisel kvaliteedikoodi kriteerium. Projektis ei kasutatud ainsatki lühendatud, mitmetähenduslikku ja arusaamatut nimetust, mis võimaldab edaspidisel arengul mitte segadusse sattuda, kui üritatakse aru saada, mida, miks ja kuidas kasutatakse, ning see on osa *"Clean Code"* põhimõttest.

Kogu projekt on jagatud kaheks osaks – serveri osaks ja kasutajaliideseks, mis on selgelt arusaadav ja nähtav tööriistaga õppides ja sellesse süvenedes. Statistikakalendri teisendamise ja õige teabe esitamise funktsionaalsuse väljatöötamisel kasutati originaalfunktsioone, mis kirjutati nullist käsitsi, ilma täiendavaid kolmandate osapoolte teeke kasutamata, välja arvatud üks – dateFns, mida kirjeldati 3.3 peatükis. Selle kasutamine hõlbustas selle meetodi rakendamist, kuna vastasel juhul tuleks erinevate kuupäevade ristumiskoha kontrolli vahepealsed meetodid rakendada käsitsi. Muidu rakendasid selles projektis osalejad kogu statistika arvutamise funktsiooni iseseisvalt. Meetodite juurutamine toimus vastavalt JavaScripti programmeerimiskeele poolt pakutavate olemasolevate funktsionaalsuste kasutamise meetodile.

Selle projekti serveriosa arendamisel kasutati nii olemasolevaid meetodeid ja funktsioone, mis töötati välja väljaspool seda projekti, kui ka nullist välja töötatud meetodeid ja funktsioone. Serveripoole arendamise alguses oli raskusi, sest ei meeskonnatöö raames ega ka selles projektis osalejate varasemas kogemuses puudus Node.js-iga seotud kogemus. See aga ei takistanud täitmast kõiki meeskonnale seatud eesmärke. Kogu funktsionaalsuse rakendamine toimus praktikas õppimise meetodil, kuna meeskond ei saanud endale lubada vajaliku materjali õppimist ilma töö edenemiseta selgelt seatud, kokkusurutud ajaraamide tõttu. Arendatud API päringud, andmete valideerimine, lõpp-punktid, andmebaasipäringud ja nendega seotud funktsioonid on täielikult kooskõlas kogu taustaprojekti üldpildiga. Koodis pole meetodi ja funktsionaalsuse üleliigseid või arusaamatuid osi – kõik on täpne ja selge.

Käesolevas töös kirjeldatud funktsionaalsuse arendamise käigus ei häiritud mitte ühegi enne meid arendatud komponendi või teenuse tööd. Uus funktsionaalsus sobitus selgelt olemasolevaga ning sai ka põhjalikult testitud nii meeskonna enda poolt käsitsi kui ka testide väljatöötamise tulemusena.

6.1.4 Testid

Kuna ettevõttes on iga arendaja ka testija, siis selle projekti elluviimisel oli testide väljatöötamine kohustuslik. Seda asjaolu silmas pidades sai testide väljatöötamine ka käesoleva töö üheks eesmärgiks.

Rakendati nii üksuse- kui ka funktsionaalseid teste, kuna projektide arendamise käigus rakendati nii üksikuid funktsioone kui ka terveid iseseisvalt olemasolevaid teenuseid. Testid realiseeriti JavaScript'i abil koos lisafunktsionaalsusega kahe raamistiku Jest ja Chai näol – mõlemat raamistikku kirjeldati 3.3. osas. Nende raamistike valikut põhjendati sellega, et need on erinevates projektides testarenduse käigus kõige enam kasutatavad teegid. Samuti kasutatakse neid raamistikke ettevõttesiseselt, mis muutis teiste teekide kasutamise peamistena võimatuks. Ülaltoodud raamistike ja JavaScript'i programmeerimiskeele kombinatsioon võimaldas nende arendamise käigus rakendada kõiki vajalikke teste.

Ühiktestide rakendamiseks deklareeriti kõik testimiseks vajalikud muutujad vajalike andmetega. Lokaalse testandmebaasi loomisel ja kasutamisel polnud mõtet, kuna töö tehti ühe funktsiooni testimisega. Ühiktestide rakendamisel mõjutati kõiki testitud funktsiooni võimalikke tulemusi. Testid töötati välja Jest raamistiku abil.

Funktsionaalsete testide rakendamiseks loodi spetsiaalselt selleks otstarbeks testide andmebaas. See andmebaas luuakse iga kord, kui funktsiooni andmeid käivitatakse. Andmebaas on täidetud spetsiaalselt loodud andmetega, millel puudub seos tegelike andmetega. Töö käigus oli võimalik välja töötada testid, mis sobivad mõlemale töö käigus rakendatavale teenusele. Seda soodustas asjaolu, et mõlemad teenused põhinevad POST API päringutel. Rakendatud test katab täielikult kogu testitava teenuse, välja arvatud osad, mis kasutavad kolmanda osapoole teenuseid. See on mitme testi kogum, millest mõned on seotud või kasutavad sama allikat. Saadud testi rakendamise käigus testiti suurt hulka võimalikke testirakendusi. Testid ja kogu nende funktsionaalsus töötati välja Chai raamistiku abil.

Kokku katavad testid 90% kogu arendatud koodist.

6.2 Kirjanduse ülevaade

Projekti väljatöötamisel järgis meeskond põhiprintsiipe, mida kirjeldas *M. Fowler*'i raamatus *„Clean Code“*, samuti *Robert C. Martin*'i raamatus *„Clean Code“*. [17], [18] Kirjutades, kui meeskond veendus, et koodi struktuur on puhas ja arusaadav, oli koodi dubleerimine minimaalne

ja kood oli arusaadav ka neile inimestele, kes polnud selle projektiga varem kokku puutunud. Kõik muutujad, meetodid ja muud koodikomponendid nimetati nii, et need annaksid edasi nende olemuse ega oleks eksitavad. Samuti on kõik funktsioonid jagatud väikesteks alammeetoditeks, et mitte ühtegi meetodit teabe ja ülesannetega üle koormata.

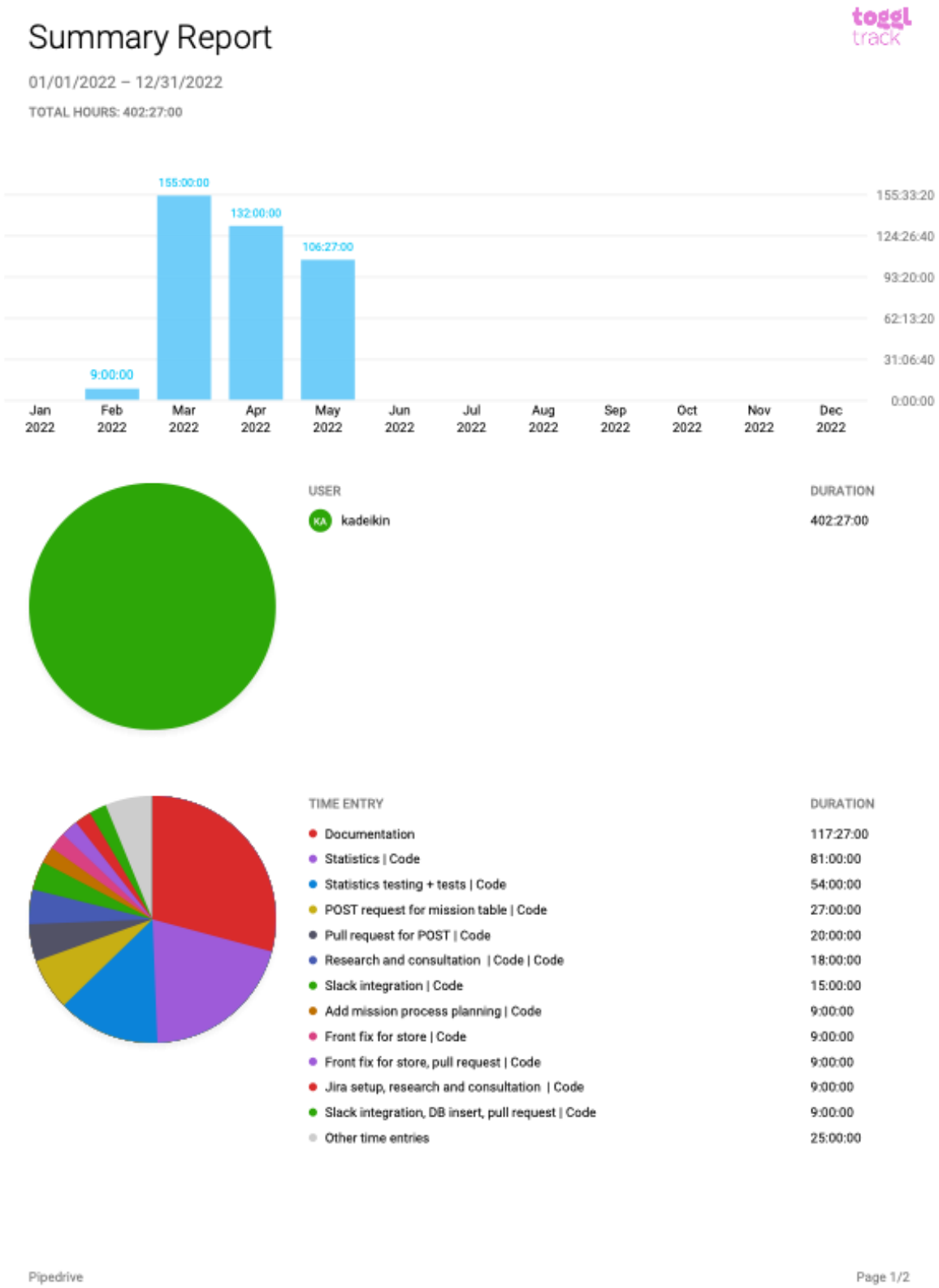
Node.js'i struktuuri ja tähenduse sügavamaks mõistmiseks on uuritud selliseid raamatuid nagu *David Herroni „Node.js Web Development"* ja *Mario Casciaro*, *Luciano Mammino„Node.js Design Patterns"* [23], [24]. Arendusse süvenedes puudutati ka mõnda osa raamatust, nagu *Ethan Brown'*i *"Web Development with Node and Express"* [25]. Kuna arendus viidi läbi süsteemidega, mida projekti väljatöötamisel osalejad polnud kordagi puudutanud, oli vaja saada mõjutatud komponentidest vähemalt esmased teadmised, millele aitasid kaasa ülaltoodud raamatud. Samuti vajas see tabamust ja rohkem professionaalseid teadmisi, kuna arendus ei olnud nii lihtne.

Testide arendamiseks vajalikud teadmised saadi *Sandro Pasquali* ja *Kevin Faaborgi* raamatu *„Mastering Node.js – Second Edition"*[26] uurimisel. Raamat kirjeldab hästi, kuidas kasutada Chai raamistikku, mida kasutati testide väljatöötamiseks. Just tänu sellele raamatule õnnestus välja töötada kõige tõhusamad testid. Näiteks see, et kahe teenuse jaoks oli võimalik ühendada vajalikud testid üheks testikogumiks, oli tänu ülaltoodud raamatule.

Algteadmised tarkvaraarendusest programmeerimiskeelt JavaScript'i kasutades saadi meistrikursuste käigus selle projekti mentoritelt, kuid mitte ilma kirjanduseta. *Douglas Crockford*'i raamat *„JavaScript: The Good Parts"* [27] oli JavaScripti programmeerimiskeelega töötamise peamine tõukejõud. Tänu sellele lühenesid paljud koodijupid märgatavalt ning hoiti ära ka arendatud funktsionaalsuse kiiruskadu.

Kogu ülaltoodud materjal leiti ja kasutati nii *O'Reilly* õpikeskkonna kui ka *Google Scholari* abil [28], [29]. Ülaltoodud väljaannete materjalide kasutamisel järgiti kõiki õigusi.

7 Teostatud tööde logi



Joonis 8. Aruanne projektiga töötamiseks kulunud aja kohta

8 Hinnang projekti teostamise protsessi kohta

See peatükk annab ülevaate valminud projektist, selle juhtimisest, koostööst ja elluviimise protsessist, ning sisaldab ka tagasisidet ettevõttelt, kus projekt välja töötati.

8.1 Projekti juhtimise ja teostamise protsess

Kuna ettevõttel paluti mitte mainida ühtegi ettevõtte töötaja nime, märgiti töös välja vaid väljatöötatava projektiga seotud ettevõtte töötajate ametikohad.

Kogu projekt töötati välja Pipedrive ettevõtte raames, milles oli meil juht – ettevõtte peainsenerijuht.

Selle projekti käigus abistasid meeskonda ettevõtte mentorid – ettevõtte ühe meeskonna insenerijuht ja *Full-Stack*'i arendaja, kes osutusid väga sõbralikuks ja meeskonda austavaks. Kogu töö tegi meeskond ise, milles mentorid täitsid ainult abi rolli, samuti kontrollisid kavandatud muudatusi/täiendusi. Mentorid olid tööajal alati ühenduses ettevõttesiseses töötajatevahelises suhtluskeskkonnas – Slackis ning vajadusel tulid meeskonna kontorisse isiklikult.

Selle projekti arendamisega alustati kohe pärast ITB1706 aine raames projektiarenduse lõppemist. Töö jätkub tänaseni ja jätkub kuni selle täieliku rakendamiseni. See projekt oli eelnevalt kokku lepitud, mis võimaldas nii meeskonnal kui ka ettevõttel selleks valmistuda. Kõik selle projekti arendamise aspektid lepiti eelnevalt kokku, määrati soovitud tulemus ja pandi paika ajaraam, mille meeskond määras enne projekti arenduse algust.

Meeskond teadis juba enne selle projekti algust, millega nad tegelevad, mida neilt nõutakse ja kuidas töövoog kulgeb, mis võimaldas meeskonnale kiiresti selle projektiga tegelema hakata. Ettevõtte andis meeskonnale arendamiseks kõik vajalikud tööriistad, seadmed, teadmised ja isegi isikliku ruumi.

Meeskond ei puutunud selle projekti arendamise ajal kokku seadistusprobleemidega, abi puudumisega, seisakutega ega konfliktidega.

Algselt koosnes meeskond kahest inimesest – Aleksei Kadeikin ja Sergei Vasiljev. Kogu projekti töö oli algselt ette nähtud jagatuna meeskonnaliikmete vahel võrdselt, mida ka tehti. Projekti kallal töötamise käigus tekkis Sergeil aga mõned probleemid, mis sundisid teda arenduses osalemise

lõpetama ja projektist lahkuma. Selle tulemusena tegi kogu selle projekti raames tehtud töö Aleksei ise.

Meeskond jätkab selle projekti arendamist tänaseni, sest ei soovi ettevõttele üle anda toodet, millel puuduvad kõik vajalikud funktsioonid sellega mugavaks töötamiseks.

Kõik nimetatud arendused selle töö käigus teostati 100% Aleksei Kadeikini poolt.

8.2 Hinnang projekti protsessile

Projekti arendamise käigus sujus kõik üsna libedalt, kuid mitte raskusteta. Arendustöö alguses oli raskusi statistika genereerimise meetodi väljatöötamisega, mis peatas korraks töö – meetodit tuli mitu korda ümber kirjutada, optimeerida, testida jne, kuid arendusest tingitud viivitus. See meetod ei olnud kriitiline. Raskuste hetkedel üritas meeskond esmalt ise hakkama saada, kuid kui see ei õnnestunud, pöörduti abi saamiseks mentorite poole. Arendus toimus samm-sammult, sest sageli sõltus mõne komponendi töö teiste tööst.

Kuna selles projektis osalejatel ei olnud palju kogemusi asjadega, mis projekti arendamise käigus mõjutasid ning mõnel juhul puudus kogemus üldse, siis kulus mõnel projektil plaanitust kauem aega. Programmeerimiskeelt, tehnoloogiaid ja raamistikke uuriti või nende kohta teadmisi täiendati vahetult töö enda käigus, kuna nende uurimiseks jäi ajapuudus. Alguses andis meeskonnatöö raske töö tunde, kuid aja jooksul astus meeskond töörütmi, tööviljakus tõusis. Teenuste juurutamine ja kõik testid läksid tõrgeteta, sest meeskonnal olid nende rakendamiseks vajalikud teadmised juba nende arendamisega alustades. Esinesid ainult väikesed raskused ja ummikud, kuid need olid väikesed. Väärib märkimist, et abiteenuste väljatöötamise ajal ei olnud praktiliselt ühtegi mentorit, mis on fakt meeskonna kasvust selle projektiga töötamise ajal.

Kui vaadata takistuste fakte projekti väljatöötamise ajal, siis peamiseks takistuseks oli dokumentatsiooni puudumine tööriista kohta, milles arendus toimus. Seda asjaolu silmas pidades pidi meeskond projekti ja selle komponentidega palju süvitsi minema, mis võttis aega, millest ilma selleta ei piisanud. See aga ei takistanud lõpuks seatud eesmärkide elluviimist. Kuna tööriist, milles arendust teostati, on juba üsna vana, 3-5 aastat vana, oli ettevõttel selle arendusperioodil uskumatult kiire kasv, mistõttu dokumentatsioon lihtsalt ununes. Nüüd, pärast vigade kallal töötamist, on kõigil ettevõtte uutel tööriistadel iga toote ja tööriista kohta korralik dokumentatsioon. Teiseks suuremaks takistuseks selle projekti arendamisel oli asjaolu, et algselt arvutati töid paarikaupa, kuid projekti käigus tegi kogu töö üks inimene. Seda asjaolu silmas

pidades selgus, et realiseeriti palju vähem, kui oli plaanis. Peamised takistused selle projekti arendamisel on loetletud eespool. Oli ka muid takistusi, kuid need olid väikesed.

Siinkohal tänab meeskond Pipedrive'i, kes andis projekti elluviimiseks kõik vajaliku ning eelkõige tõstab esile ettevõtte poolt kaasa antud mentorid, kes meeskonda alati aitasid ega jätnud hätta.

Selle projekti arendamine jätkub tänaseni ja jätkub kuni kõigi kavandatud ideede elluviimise lõpuni.

8.3 Hinnang projekti teostamisele

Meeskond usub, et projekt õnnestus – kõik oli planeerimisetapis hästi planeeritud, meeskond ei viivitanud projekti käivitamisega ning arendusprotsessi käigus püüdis meeskond mitte ühel kohal seista. Meeskond oli alati ettevõtte ja mentoritega ühenduses, ei kartnud nendega ühendust võtta ning näitas pidevalt ka oma töö tulemusi. Projekti kallal töötamise käigus on meeskond oma teadmisi kõvasti laiendanud ja on oma saavutuste üle uhke.

Kõik tööd tehti nii selleks ette nähtud kontoris, kui olukord seda võimaldas, kui ka distantsil. Kui kontoris töötades probleeme ei esinenud, siis projekti kaugarendamisel tekkisid meeskonnal ebamugavused üksteise üle kontrolli puudumise näol – selle probleemi tagajärjeks oli ühe projektis osaleja kadumine. Kui projekti kaugarenduse käigus oli vaja suhelda, kasutas meeskond ettevõttesisest suhtluskeskkonda – Slack'i, aga ka Zoom'i.

Projekti arendamise käigus oli nõrgim külg ühe osaleja väljalangemine, mis sundis muutma tööplaani ja ümber arvestama kõigi eesmärkide täitmiseks vajalikku aega. Mõned eesmärgid tuli üldse tühistada, kuna projekti arendamiseks oli aeg piiratud.

Kõik selle töö raames seatud eesmärgid täitis Aleksei osaliselt või täielikult, kuna projekti selles konkreetses osas osaleja ei saanud tööd jätkata.

8.4 Ettevõttepoolne hinnang meeskonna tööle

Selle projekti arendus viidi läbi Pipedrive'i projekti osana. Ettevõte jäi tulemusega rahule, kuigi projekt ei jõudnud määratud ajaks valmis. Ettevõttesisene projektijuht rõhutas, et kõik selle projekti raames tehtavad tööd tehti ilma ettevõtte arendajate osaluseta, kuid meeskonnaga käis kaasas vaid mentorite abi keeruliste ülesannete, probleemide või küsimuste lahendamisel. Mentorid märkisid, et projekti edenedes pidid nad iga kord vähem meeskonda aidata, kuna

meeskond saaksid neile pandud ülesannetega ise hakkama. See annab märku, et meeskonnaliikmed omandasid projekti käigus uusi teadmisi ja kasutasid neid ka praktikas.

Mentorid andsid ka tagasisidet iga selle projekti elluviimisel töötanud meeskonnaliikme kohta.

8.4.1 Tagasiside Aleksei Kadeikini tööle

Mentoritelt saadud tagasiside Aleksei Kadeikini töö kohta:

"Tarkvara kasutajaliidese täiendamise eesmärgiks oli luua kasutajatele süsteem lihtsustamaks planeerimist eesootavate tööde jaoks.

Tallinna Tehnikaülikooli tudeng Aleksei Kadeikin arendas antud töö raames välja back-end osa planeerimissüsteemi toimimiseks.

Arenduse käigus näitas üliõpilane suurepäraseid teadmisi ning oskusi hakkama saada keerukate ja andmemahukate ülesannete lahendamisel.

Samuti disainis ja implementeeris üliõpilane uusi andmetabeleid ning aitas kaastudengeid erinevate probleemide lahendamisel.

Kuigi antud projekti läbiviimise aeg oli lühike ning eelnev kokkupuude JavaScript'iga oli vähene, panustas üliõpilane arendatud rakenduse valmimisse märgatavalt." – Pipedrive'i mentorid

9 Hinnang meeskonnaliikmete panusele

Ettevõttes tulevaste projektide planeerimise lahenduse väljatöötamise projekti elluviimisel näidatakse iga osaleja panust hindega “-1” kuni “+1”, kus “-1” on halvim punktisumma. Halvim hinnang tähendab täielikku panuse puudumist arendatud projekti.

Meeskond otsustas punktid jagada järgmiselt:

- Aleksei Kadeikin: “+1” (panustas projekti maksimaalselt),
- Sergei Vasiljev: “-1” (projekti panustamine oli olenematu).

10 Kokkuvõte

Projekti eesmärk oli rakendada Pipedrive'is tulevane projektigraafiku lahendus. Peamine probleem, mida väljatöötatud lahendus aitab lahendada, on hõlbustada ettevõtte meeskondade seisu jälgimist, aga ka uute projektide loomist. Projekti eesmärgiks oli välja töötada lahendus, mis annab täieliku statistika iga meeskonna töötajate tööhõive kohta, samuti võimaluse luua väljatöötatud lahenduse raames uusi projekte ning kutsuda töötajaid projektidesse.

Võrreldes ITB1706 aine raames välja töötatud meeskonnaprojekti aegse tööga, oli meeskonna töö selle töö raames projekti väljatöötamisel palju ühtsem, kiirem ja selgem. Märkimist väärib ettevõtte ja mentorite meeskonna tugev toetus, mis aitas kaasa selle projektiga tehtud töö positiivse tulemuse saavutamisele. Meeskond sai hea kogemuse reaalse ja suure projekti kallal töötamisest, mis lõpuks õnnestus ka ellu viia. Esmase kogemusena võib kogemuse kindlasti lugeda positiivseks ja õnnestunuks.

Tehtud töö tulemuseks oli lahendus, mis annab kasutajatele täieliku ülevaate valitud meeskonna statistikast, selle staatusest, aga ka käimasolevate ja kavandatavate projektide seisust. Realiseeriti nii uue projekti loomise kui ka inseneride kutsumise võimalus olemasolevatesse projektidesse. Osa funktsionaalsust on veel väljatöötamisel, kuid lahendus juba töötab ja seda saab kasutada.

Tehtud tööd analüüsides on meeskond kindel, et lahendus on mugav ja praktiline, selle kasutamine toob kasu nii töötajatele kui ka ettevõttele tervikuna. Kuigi meeskonnal tekkis lahenduse väljatöötamisel nii suuri kui ka väiksemaid raskusi, võib projekti lugeda kordaläinuks. Meeskond tunneb uhkust selle üle, et valis projekti elluviimise planeerimise etapis õiged meetodid ja järgis arenduse käigus täpselt määratletud plaani.

Kasutatud kirjandus

[1] Pipedrive, 2022. [Online]. Loetud aadressil: https://www.pipedrive.com/about
Kasutatud: 28.04.2022

[2] Slack, 2022. [Online]. Loetud aadressil: https://slack.com/help/articles/115004071768-What-is-Slack-
Kasutatud 28.04.2022

[3] Confluence, 2022. [Online]. Loetud aadressil:
https://www.atlassian.com/software/confluence/guides/get-started/confluence-overview#hosting-options
Kasutatud: 28.04.2022

[4] Airtable, 2022. [Online]. Loetud aadressil: https://www.airtable.com/about
Kasutatud: 28.04.2022

[5] Grafana, 2022. [Online]. Loetud aadressil: https://grafana.com/docs/grafana/latest/introduction/
Kasutatud: 28.04.2022

[6] BambooHR, 2022. [Online]. Loetud aadressil: https://www.bamboohr.com/about/
Kasutatud: 28.04.2022

[7] Annabel Matkur, Lii Saluvere, Ettevõttesisese tarkvara täiendamine – kasutajaliidese ja andmebaasi kavandamine ning sobivuse hindamine, Tallinn: TTÜ kirjandus, 2022

[8] Vue.js Releases. [Online]. Loetud aadressil: https://vuejs.org/about/releases.html
Kasutatud 29.04.2022

[9] Visual Studio Code. [Online]. Loetud aadressil: https://code.visualstudio.com/docs
Kasutatud: 29.04.2022

[10] GitHub. [Online]. Loetud aadressil: https://github.com/about
Kasutatud: 29.04.2022

[11] Docker. [Online]. Loetud aadressil: https://docs.docker.com/get-started/overview/
Kasutatud: 29.04.2022

[12] Jira. [Online]. Loetud aadressil: https://www.atlassian.com/software/jira/guides/getting-started/overview
Kasutatud: 29.04.2022

[13] Node.js. [Online]. Loetud aadressil: https://nodejs.org/en/about/
Kasutatud: 01.05.2022

[14] Olga Filipova, Learning Vue.js 2. [E-Book]. Loetud aadressil:
https://books.google.ee/books?hl=en&lr=&id=nszcDgAAQBAJ&oi=fnd&pg=PP1&dq=Vue.js+history&ots=9oJdEiSepH&sig=C59H_FNBdp4Brty5Z_gWl7fyY0g&redir_esc=y#v=onepage&q=Vue.js%20history&f=false
Kasutatud: 03.03.2022

[15] Mariliis Sinivee, Reia Rõõmus, “Ettevõttesisese tarkvara kasutajaliidese täiendamine”, Tallinn: TTÜ kirjandus, 2022

[16] Zoom. [Online]. Loetud aadressil: https://explore.zoom.us/en/about/
Kasutatud: 01.05.2022

[17] C. Larman, Applying UML and Patterns, 2004. [E-Book]. Loetud aadressil: https://www.oreilly.com/library/view/applying-uml-and/0131489062/
Kasutatud: 07.03.2022

[18] R. Harmes, Pro JavaScript™ Design Patterns, 2007. [E-Book]. Loetud aadressil: https://www.oreilly.com/library/view/pro-javascripttm-design/9781590599082/
Kasutatud: 09.03.2022

[19] SonarQube. [Online]. Loetud aadressil: https://www.sonarqube.org/about/
Kasutatud: 15.03.2022

[20] CodeShip. [Online]. Loetud aadressil: https://www.cloudbees.com/products/codeship
Kasutatud: 15.03.2022

[21] Jest. [Online]. Loetud aadressil: https://jestjs.io/
Kasutatud: 15.03.2022

[22] Chai. [Online]. Loetud aadressil: https://www.chaijs.com/
Kasutatud: 15.03.2022

[23] D. Herron, Node.js Web Development, 2018. [E-Book]. Loetud aadressil: https://books.google.ee/books?hl=en&lr=&id=dnteDwAAQBAJ&oi=fnd&pg=PP1&dq=Node.js+Web+Development+david&ots=_OyRaUTfxB&sig=4HcfxZ_AwrBjN_MopyaY-4Ln7ws&redir_esc=y#v=onepage&q=Node.js%20Web%20Development%20david&f=false
Kasutatud: 10.03.2022

[24] M. Casciaro, L. Mammino, Node.js Design Patterns, 2016. [E-Book]. Loetud aadressil: https://books.google.ee/books?id=55WqDQAAQBAJ&printsec=frontcover&dq=Mario+Casciaro+%E2%80%9ENode.js+Design+Patterns%E2%80%9C.&hl=en&sa=X&ved=2ahUKEwiGvP3goOf3AhVPxIsKHXcTA4kQ6AF6BAgFEAI#v=onepage&q=Mario%20Casciaro%20%E2%80%9ENode.js%20Design%20Patterns%E2%80%9C.&f=false
Kasutatud: 10.03.2022

[25] E. Brown, Web Development with Node and Express, 2014. [E-Book]. Loetud aadressil: https://books.google.ee/books?id=1cHvAwAAQBAJ&printsec=frontcover&dq=Ethan+Brown%E2%80%99i+%E2%80%9CWeb+Development+with+Node+and+Express%E2%80%9D.&hl=en&sa=X&ved=2ahUKEwjsxt2foef3AhXE-ioKHflFCT0Q6AF6BAgEEAI#v=onepage&q=Ethan%20Brown%E2%80%99i%20%E2%80%9CWeb%20Development%20with%20Node%20and%20Express%E2%80%9D.&f=false
Kasutatud: 01.04.2022

[26] S. Pasquali, K. Faaborg, Mastering Node.js - Second Edition, 2017. [E-Book]. Loetud aadressil: https://www.oreilly.com/library/view/mastering-nodejs/9781785888960/
Kasutatud: 12.04.2022

[27] D. Crockford, JavaScript: The Good Parts: The Good Parts, 2008. [E-Book]. Loetud aadressil: https://books.google.ee/books?id=PXa2bby0oQ0C&printsec=frontcover&dq=Douglas+Crockfordi+raamat+JavaScript:+The+Good+Parts&hl=en&sa=X&ved=2ahUKEwiA5Kukouf3AhWql4sKHXVtDzsQ6AF6BAgDEAI#v=onepage&q&f=false
Kasutatud: 08.03.2022

[28] O'Reilly. [Online]. Loetud aadressil: https://www.oreilly.com/about/
Kasutatud: 17.04.2022

[29] Google Scholar. [Online]. Loetud aadressil: https://scholar.google.com/intl/en/scholar/about.html
Kasutatud: 17.04.2022

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

Mina, Aleksei Kadeikin

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Ettevõttesisese tarkvara serveripoolse rakenduse täiendamine", mille juhendaja on Jelena Vendelin
    1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
    1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

18.05.2021

[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Üksiktesti näide

```
Describe.each(statData.statisticsSource)('Getting correct statistic value', (statVal) => {
            it(`${statVal.name} test`, () => {
                  const request = {
                        stat: data.statistics[1],
                        missions: data.missions,
                        tribePeople: data.tribePeople,
                        stats: data.statValues,
                        sIndex: 1,
                  };
                  const result = utils.getCorrectStatValue(request, statVal);
                  expect(result).toBe(correctValues[statVal.name]);
            });
      });
```

Lisa 3 – Funktsionaalse testi näide

```
[{data1},{data2}].forEach((values) =>
      describe(`mission ${values.method} request...`, () => {
            const options = {
                  method: ’POST’,
                  url: values.url,
                  headers: {
                        Authorization: …,
                  },
                  json: true,
                  data: values.data,
            };
            const whereValues = values.multiple ? coll1 : coll2;
            const deleteTestData = async () => {
                  await ctx
                        .db(values.table)
                        .delete()
                        .where(...whereValues);
            };
            it(`won’t post without all required data`, async () => {
                  await axios(options).catch((err) => {
                        expect(err.response.status).to.equal(400);
                        values.requiredFields.forEach((x) =>

      expect(err.response.data.error.message).to.contain(`“${x}“is
required`),
                        );
                  });
            });
            afterEach(async () => {
                  await deleteTestData();
            });
      }),
);
```

Lisa 4 – Aleksei Kadeikini eneseanalüüs

Projektide planeerimist hõlbustava lahenduse väljatöötamisega alustasin novembris meeskonnaprojekti raames. Kogu selle projekti arenduse aja olin ma aktiivne meeskonnaliige ja olin pidevalt arendusprotsessis kaasas ning olin kursis projekti seisuga.

Kohe pärast meeskonnaprojekti kaitsmist asusime seda lahendust edasi arendama, kuid meeskond oli esialgu suurem. Esimestel nädalatel pärast meeskonnaprojekti kaitsmist kohtusime ettevõttesisese juhiga, arutasime tulemusi, tugevaid ja nõrku külgi, arutasime edasist plaani ja ajakava. Seejärel alustas tegevust uue projekti analüüsi ja prototüübi eest vastutav meeskond, ülejäänud meeskond aga otsustas meeskonnaprojekti raames tegemata jäänud asjad lõpuni viia. Peamine raskus oli asjaolu, et mõlema projekti elluviimisel lähtuti asjadest, mida ülikoolis õppimise ajal vähe või üldse ei õpitud, samas oli meeskonnal kõrge motivatsioon, mis aitas sellest raskusest üle saada.

Kuna kogu projekt seisnes uue tööriista väljatöötamises, mis kuulub Mission Tooli lahenduse üldkollektsiooni, siis otsustati selles tööriistas kasutada nii olemasolevaid arendusi kui ka nullist uute loomist. Projekt jagunes 3 osaks - prototüübi analüüs ja arendus, kasutajaliidese arendus ja serveripoole arendus. Hakkasime Sergeiga serveriosa hõivama. Algusest peale alustasin tööd iseseisvalt, sest Sergeil oli raskusi, mis ei võimaldanud tal algusest peale projektis osaleda. Mõne aja pärast hakkas ta siiski projekti kallal töötama, kuid mitte kaua. Kui Sergei täielikult kadus, otsustasin selle projekti ise teha. Kuna mul oli algselt JavaScript'i programmeerimiskeelega mingi kogemus ja tänu meeskonnaprojektiga töötamisel saadud kogemustele ei olnud mul raske aru saada, mida on vaja taustasüsteemis teha, kuidas seda teha. ja milleks seda kasutatakse. Kõik Node.js'iga seotud teadmised osutusid aga ebapiisavaks, mistõttu pidin seda teemat mõnda aega uurima. Selleks kasutasin kirjandust ja projektis juba olemasolevaid komponente ning mentorite abi. Alustamine oli keeruline ka seetõttu, et tööriistal puudus dokumentatsioon, mistõttu projektiga tutvumine võttis algselt planeeritust rohkem aega.

Uue projekti jaoks vajalike komponentide väljatöötamist alustades takistusi ei olnud, kuid keset arendust põrkasin kokku väikese probleemiga - ülesanne, millega tegelesin, osutus palju keerulisemaks, kui varem tundus, mis tähendab, et selle lõpliku väljatöötamiseni kulus rohkem aega. Sellest hetkest hakkasin seatud ajaraamidest veidi maha jääma. Siinkohal palusin abi mentoritelt, kuna läksin arenduse käigus täiesti segadusse, kuid kõik osutus üsna lihtsaks ja sain turvaliselt edasi töötada. Kui jõudsin selleni, et taotlesin enda väljatöötatud koodi ülevaatamist, leidsin natuke aega, kuid mitte kauaks. Minu pakutud koodi kontrollides pakkusid mentorid välja

palju parandusi ja muudatusi, milleni ma ise ei jõudnud. Seda silmas pidades pidin kogu väljatöötatud koodi sõna otseses mõttes nullist üle vaatama. Pärast seda kordus jutt veel mitu korda, kuni saadi ideaalne variant. Tänu nendele kontrollidele õppisin palju, kuid jäin graafikust maas.

Peale seda asusin tegelema kahe ülesandega - osalejate kutsumise süsteemi juurutamine ja uute projektide loomise süsteem läbi arendatava tööriista. Nende süsteemide kallal töötamise käigus mul probleeme praktiliselt ei olnud, kuna teadmised olid selleks hetkeks juba piisavad. See võimaldas mul välja töötada piisavalt kvaliteetseid süsteeme, mida mentor nende kontrollimise etapis väga kiitis. Samuti tõi mentor nende kontrollimise etapis esile mitu punkti, mis teda rõõmustasid, kuid oli ka nõrkusi - nende lahendamine ei võtnud palju aega.

Testide väljatöötamisega mul praktiliselt probleeme ei olnud, mis võimaldas mul neile mitte palju aega kulutada. Sellele aitasid kaasa ülikooliõpingute ajal saadud head teadmised, sest ka siis polnud mul kontrolltöödega suuri probleeme. Mul õnnestus olemasolevaid teadmisi võimalikult tõhusalt kasutada ning testide väljatöötamiseks tuli vaid tutvuda tööriistas kasutatava raamistikuga.

Kuna teadsin hästi ka Vue.js'il põhineva kasutajaliidese arendamise teemat, siis aitasin selle arenduse käigus sellega tegelenud meeskonda rohkem kui korra.

Võin kindlalt öelda, et selle projekti väljatöötamise juures ei olnud kõige keerulisem mitte projekti enda väljatöötamine, vaid projektis juba olemasolevate lahenduste uurimine. Mul pole kunagi varem olnud kogemusi ühegi suure projekti kallal, veel vähem sellise mahuga projektiga ja isegi ilma dokumentatsioonita. Natuke sandistas mind ka see, et sõna otseses mõttes keset arengut, õigemini veidi hiljem katkes kontakt Sergeiga täielikult, mille tulemusena sain aru, et selle töö edasine tulemus sõltub ainult minust endast. Probleemiks ei olnud see, et ma üksi jäin, vaid see, et algselt oli töö arvestatud kahe inimese ja meeskonnatööga ning selle tulemusena jäin üksi. Kuid see asjaolu oli ka positiivne, sest tänu sellele sain kogeda kõiki võimalikke tulemusi päriselus, samuti töötada suure hulga komponentide ja materjalidega.

Selle projekti lõpus võin julgelt öelda selle eduka lõpuleviimise kohta. Jah, kõik planeeritud asjad pole välja töötatud, kuid enamik neist on välja töötatud. Hea meel oli ka selle üle, et kõik takistused tulemuste saavutamisel said ületatud ning õppisin palju.

Olen tohutult tänulik kogu meeskonnale, kes selle projekti kallal töötas, selle arendamise käigus saadud kogemuste eest, aga ka mentoritele ja Pipedrive'ile, kelleta poleks seda kõike juhtunud.