TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Henri Haugas 213660IABB

Kadri-Liis Kolk 213625IABB

Kohila pikamaajooksu sarja registreerunute haldamise ja jooksude komplekteerimise rakendus

Bakalaureusetöö

Juhendaja: Tarvo Treier

MSc

Tallinn 2024

Autorideklaratsioon

Kinnitame, et oleme 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.

Autorid: Kadri-Liis Kolk, Henri Haugas

20.05.2024

Annotatsioon

Spordivõistluste korraldamine nõuab sageli mahukat ja keerukat andmete haldamist, eriti kui tegemist on suure hulga osalejatega. Osalejate andmete käsitsi haldamine ja nende komplekteerimine võib osutuda äärmiselt ajamahukaks ja vigaderohkeks. Digitaalsete lahenduste kasutamine pakub tõhusat alternatiivi, automatiseerides ja optimeerides kogu protsessi. Sellist lahendust vajas Kohila pikamaajooksu sari, et vähendada ajakulu, hõlbustada registreerunute haldamist, jooksude komplekteerimist ja tulemuste väljastamist.

Käesoleva töö eesmärgiks on välja töötada rakendus, mis võimaldab tõhusalt hallata Kohila pikamaajooksu sarja võistlejaid ja jooksude komplekteerimist ning vähendaks käsitsi andmete ümberkirjutamist. Töö käigus analüüsivad autorid probleemi olemust ning erinevaid olemasolevaid lahendusi ja tööriistu, mis toetavad registreerimise ja jooksude komplekteerimise funktsionaalsusi.

Töö raames kogutakse vajalikud nõuded Kohila pikamaajooksu sarja korraldajatelt ning rakendatakse tarkvaraarenduse parimaid tavasid. Analüüsitakse lahenduse vastavust kogutud nõuetele ning kirjeldatakse valikuid, muudatusi ja üldist arendusprotsessi. Lahenduse funktsionaalsus valideeritakse nii kliendi kui ka osalejate poolt ning dokumenteeritakse võimalikud edasiarendused tulevikus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 6 peatükki, 20 joonist, 2 tabelit.

Abstract

Kohila long distance running series registration management and run completion application

Organizing sports competitions often requires extensive and complex data management, especially when dealing with a large number of participants. Manually managing and grouping participant data can be extremely time-consuming and error prone. The use of digital solutions offers an efficient alternative by automating and improving the entire process. Such a solution was needed by the Kohila long distance running series to reduce time consumption, ease the management of registrations, the grouping of runs and the release of results.

This work aims to develop an application that will enable effective management of Kohila long-distance running series competitors, race set-up and reduce manual data rewriting. In the course of the work, the authors analyze the nature of the problem, various existing solutions and tools that support the functionalities of registration and assembly of runs.

As part of the work, the requirements will be collected from the organizers of the Kohila long-distance running series and the best practices of software development will be implemented. The solution's compliance with the collected requirements is analyzed, and the choices, changes, and overall development process are described. The functionality of the result is validated by both the client and the participants, and potential future developments are documented.

The thesis is written Estonian and contains 40 pages of text, 6 chapters, 20 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	Application Programming Interface ehk liides rakenduse loogikale
Front-end	Kasutajaliides
Back-end	API koos loogikaga ja kasutajaliidese andmevahetusest
CSS	*Cascading Style Sheets*, kaskaad-stiililehed, küljendamisel kasutatav märgistuskeel
React	JavaScripti teek kasutajaliidese loomiseks
HTTP meetod	Päringutüüp
JavaScript	Programmeerimiskeel
Node.js	Serveripoolsete rakenduste arendamise keskkond
Express.js	Node.js-i raamistik veebirakendustele
MySQL	Andmebaasisüsteem
Axios	JavaScriptis kirjutatud HTTP klient
JSON	JavaScript *object notion* ehk andmevahetusel kasutatav vorming vorming
HTML	*HyperText Markup Language* on veebilehtede struktureerimiseks ja sisu defineerimiseks kasutatav märgenduskeel.
SaaS	*Software as a Service*, tarkvara teenusena, pilveandmetöötluse vorm
SPA	Üheleheline rakendus, S*ingle Page Application*
MVC	Mudel-vaade-kontroller, *Model-View-Controller*
Postman	Rakendus API päringute testimiseks
UI/UX	Kasutajaliides/kasutajakogemus

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesolevas peatükis annavad töö autorid ülevaate projekti taustast, sõnastavad probleemi ning püstitavad töö eesmärgi. Lisaks tuuakse välja projekti kõige algsemad funktsionaalsed ja mittefunktsionaalsed nõuded.

1.1 Taust

Jooksuvõistluste populaarsus on viimastel aastatel pidevas tõusujärgus nii kergejõustiklaste kui ka harrastajate seas ning 2020. aastal [1] alanud koroonapandeemia tagajärjel suurenes jooksjate arv veelgi. Inimestel tekkis rohkem vaba aega ja tahet oma füüsilise tervise eest hoolt kanda, kuid spordiasutused olid kinni ning seetõttu hakati justnimelt jooksmist harrastama. Samamoodi nagu jooksmine üleüldiselt, on populaarsust kogunud ka Kohila pikamaajooksu sari, mis sai alguse 2010. aastal. Selle raames toimus 6 etappi ning kogu hooaja vältel osales punktitabeli andmetel sarjas 44 erinevat võistlejat ja kõige rohkem osales ühel etapil 23 sportlast [2]. Jooksusarja algne eesmärk oli pakkuda Kohila elanikele võimalust osaleda mitmesugustel jooksuvõistlustel ja edendada kommuuni sportlikku vaimu.

Sari hakkas kiirelt populaarsust kogunema ka harrastajate ja professionaalsete sportlaste seas väljaspool Kohilat, kuna pakub igal tasemel sportlasele konkurentsitihedat võistlust ning toob kokku erineva tasemega osalejaid, alustades algajatest kuni kogenud jooksjateni.

Tänaseks kuulub sarja 9 erineva distantsiga etappi, mis toimuvad maist augustini kolmapäeviti. Iga etapiga on võimalik koguda oma vanuseklassis punkte. Esimene koht saab 12 punkti, teine 10 punkti, kolmas 8 punkti. Kolmandast kohale järgnevad 1 punkti vahega nii, et kümnes koht annab 1 punkti. Kui ei õnnestu kümne parima sekka pääseda, saab 0,5 punkti osalemise eest. Sarjas osaleja lõppskoor arvutatakse kuue parima etapi tulemuste pealt.

Suurima osalejate arvu saavutas jooksusari 2020. aastal, peale esimest koroonapandeemiat. Kokku osales nimetatud aastal 207 erinevat sportlast ning kõige populaarsem oli 1500m etapp, kus osales 95 sportlast. Sellele järgnevatel aastatel on osalejate arv sarjas püsinud alati üle 140 erineva inimese ja aastal 2023 üle 170. [2]

1.2 Probleem, projekti eesmärk

Kohila pikamaajooksu sarja osalejate arv on üldjoontes kasvanud ning seega on käsitsi hallatud registreerimise protsess osutunud ebatõhusaks ja ajakulukaks. Etapile registreerimiseks peab võistleja saatma meili või helistama Kohila Spordikeskusele (edaspidi Kohila), mille järel Kohila administraator sisestab andmed käsitsi MS Excel programmi. Jooksude komplekteerimiseks peab administraator nimesid Excelis ringi tõstma ja kohandama vastavalt jooksjate eeldatavale ajale ning lõpuks tulemused käsitsi protokolli lisama. Käesoleva töö eesmärgiks on luua tõhus, automatiseeritud ja kasutajasõbralik süsteem, mis võimaldaks Kohila pikamaajooksu sarja registreerunute haldamist, jooksude komplekteerimist, tulemuste sisestamist ja eksportimist.

Eesmärgi täitmiseks analüüsivad autorid lähemalt olemasolevaid lahendusi, praeguseid väljakutseid ja murekohti, kaardistavad vajadused ja loovad lahenduse, mis vastab nii osalejate kui ka korraldajate ootustele. Sealjuures pööratakse tähelepanu süsteemi kasutusmugavusele ning võimalusele seda tulevikus vastavalt vajadusele laiendada. Lisaks uuritakse, kas sarnast süsteemi on võimalik rakendada ka teistele üritustele, olgu selleks jooksuvõistlused või näiteks jalgpallimatšid, tehes seejuures minimaalselt muudatusi.

1.3 Funktsionaalsed ja mittefunktsionaalsed nõuded

Järgnevalt on toodud algsed laialdased nõuded, mis said kokku lepitud kooskõlas kliendiga esimesel kohtumisel. Täpsustatud nõuded on toodud peatükis 4.1.

Funktsionaalsed nõuded:

- Võistlejate registreerimine: Süsteem võimaldab nii kasutajatel kui ka administraatoril registreerida jooksjat valitud etapile.
- Üritusega seotud toimingud: Nii kasutajatel kui ka administraatoril on võimalik vaadata lähenevaid etappe ja etappidele registreerunuid.

- Jooksjate gruppidesse komplekteerimine: Administraatoril on võimalus komplekteerida automaatselt jooksjate grupid vastavalt eeldatavatele distantsi läbimise aegadele ja jooksjate arvule, mis peaks olema ühes grupis.
- Tulemuste haldamine: Administraatoril on võimalus sisestada jooksjate tulemusi, kuvada tulemusi üldises ja vanusegruppide arvestuses ning eksportida tulemusprotokolle vastavalt vajadusele.

Mittefunktsionaalsed nõuded:

- Kasutajaliides: Süsteemi kasutajaliides on võimlikult lihtne ja kasutajasõbralik, jälgides ühtlasi Kohila Spordikeskuse kodulehe kasutajaliidese disaini.
- Skaleeritavus: Süsteem on tulevikus lihtsasti laiendatav, kohandatav ja täiendatav vastavalt kliendi vajadustele.

1.4 Töö edasine struktuur

Projekti edasine struktuur hõlmab ülevaadet olemasolevatest lahendustest ja nende võrdlusest. Tuuakse välja, millised on nende puudused ning milliseid metoodikaid saaks uue projekti jaoks üle võtta. Sellele järgneb metoodika osa, mis sisaldab arendatud süsteemi kirjeldust ning annab ülevaate kasutatud tööriistadest ja tööprotsessist. Seejärel keskendutakse töö tulemustele, sealhulgas nõuetele, arhitektuurile, disainile, koodile ja testidele. Analüüsi ja järelduste peatükis keskendutakse eelkõige projekti teostuse ning töö tulemuste põhjendusele, tulemuse valideerimisele ja võimalikele edasiarendustele. Viimaks tehakse protsessi kokkuvõtte ja esitatakse lisadena mõlema meeskonnaliikme aruanne nende panusest ja eneseanalüüsist.

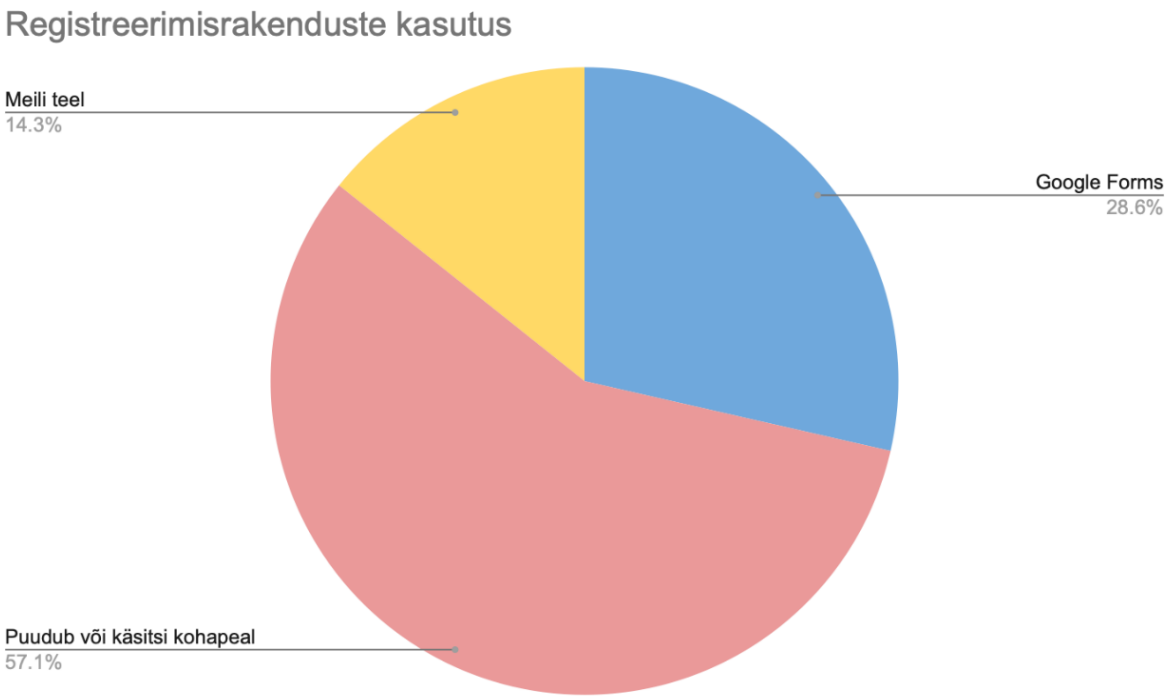
2 Ülevaade olemasolevatest lahendustest

Käesolevas peatükis antakse ülevaade olemasolevatest lahendustest, nende funktsionaalsustest ja puudujääkidest. Autorid analüüsivad, milliseid varem kasutatud meetodeid on võimalik rakendada teostatavas projektis.

Spordiürituste registreerimise ja tulemuste haldamise protsess on mitmekesine ja sõltub üldiselt konkreetsest võistlusetüübist ning selle ulatusest. Käesoleva projekti raames uurisid töö autorid erinevatelt kergejõustiku ürituste korraldajatelt Eestis, millised on nende kasutatavad lahendused, kui automatiseeritud on protsess ning kas nad on tundnud puudust süsteemist, mis ühendaks nii registreerimise, komplekteerimise kui ka tulemuste haldamise (Lisa 4). Vastuseid laekus seitsme ürituse korralduse kohta: Tartu KEVEK ja Tartu KALEV staadionijooksud, Nõmme KJK lahtised meistrivõistlused, Tallinna noorte meistrivõistlused, Nõmme Spordiklubi laste meistrivõistlused, Uno Palu noorte mitmevõistluse karikavõistlused, Tartu Valla Jooksusari ja Viljandi valla seeriajooks.

Saadud vastustest selgus, et kasutatavaid meetodeid on erinevaid, kuid mitmes vastuses toodi välja Google Forms keskkonda ning MS Excelit. Jooksude komplekteerimist teostatakse üldjuhul käsitsi MS Excelis või paberil. Võib öelda, et mitmete spordiorganisatsioonide registreerimise ja komplekteerimise protsessid on manuaalsed ning pigem ajakulukad. Samas toodi ühes organisatsioonis välja rakendust Sportity, mis kujutab endast küllaltki kaasaegset programmi, kus võistlejad saavad enda tulemusi reaalajas jälgida.

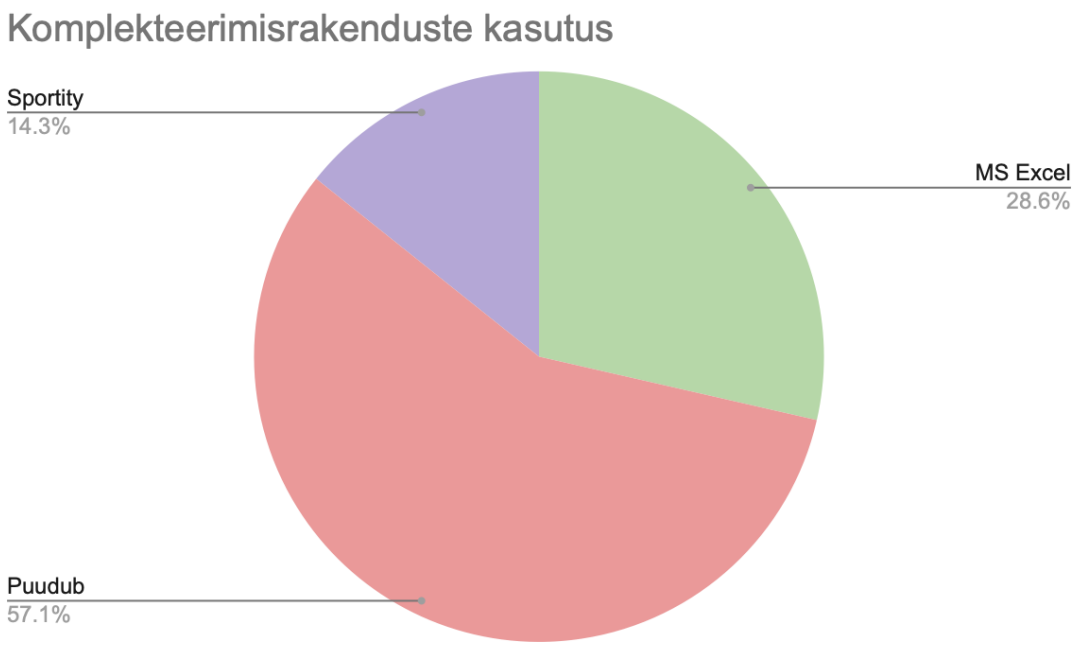
Vastuste statistika näitas, et registreerimisrakendus sageli isegi puudub ning üldjuhul registreerivad võistlejad end jooksule kohapeal (Joonis 1). Kõige rohkem mainiti registreerimisrakendustest Google Formsi (28,6%) ning seejärel meili teel registreerimist (14,3%).



Joonis 1. Registreerimisrakenduste kasutus.

Kuigi Google Forms täidab edukalt kõige primaarsemat registreerimise funktsionaalsust, leidub sellel siiski puuduseid, mis muudavad protsessi ressursikulukamaks. Vaadates Google Formsi kasutavate vastajate registreerimisvorme, võib märgata näiteks valideerimise puudulikkust, mis omakorda võib viia selleni, et kasutaja saab sisestada ebakorrektset infot või jätta välju hoopiski tühjaks. Ühe näite puhul puudus võimalus sugu valida, mis muudab kindlasti jooksude grupeerimise samuti aeganõudvamaks, kuna võistleja soo kindlaks tegemine interneti teel võtab oma osa ajast.

Komplekteerimisrakenduste kasutust analüüsides selgus samuti, et üldjuhul puudub spetsiaalne platvorm jooksude grupeerimiseks ning jooksud koostatakse taaskord käsitsi (Joonis 2). Kõige enam leiab komplekteerimise eesmärgil kasutust MS Excel (28,6%) ning teisena Sportity (14,3%).

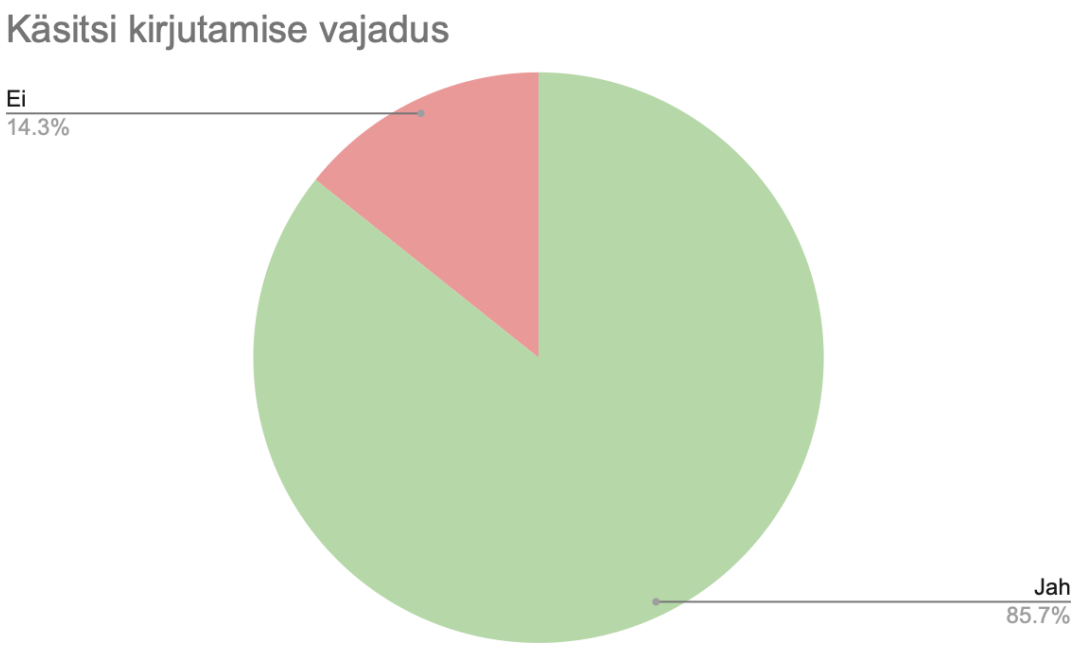


Joonis 2. Komplekteerimisrakenduste kasutus.

Sportity rakenduse kasuks räägib võimalus jookse komplekteerida ning tulemusi sisestada ja hallata. Lisaks on seal võimalik jälgida tulemusi reaalajas. Sportity oluliseks puuduseks on selle kõrge hind, mis on sageli ka peamine põhjus, miks võistluste korraldajad antud lahendust ei kasuta. Ühe võistluse korraldamise hinnaks on Sportitys toodud 50€, mis igal etapil ei mahu eelarvesse [3], sest üldjuhul ei piirdu spordiürituse korraldamine n-ö ühe üritusega, kuna jooksuseeria koosneb rohkem kui ühest etapist.

Tulemuste sisestamise ja haldamise kohta otsest küsimust töö autorid ei koostanud, kuid seda mainisid pea kõik intervjueeritavad, et tulemused sisestatakse käsitsi nii süsteemi kui ka protokolli. Sportity puhul tuleb tulemused sisestada manuaalselt süsteemi, aga protokoll koostatakse automaatselt ja saadetakse Eesti Kergejõustikuliidule.

Vastustest selgus, et kogu registreerimise, komplekteerimise ja punktihalduse protsessi jooksul peab 85,7% ürituste korraldajatest andmeid ühest kohast teise manuaalselt ümber kirjutama ning seda eelkõige just tulemusi protokolli sisestades, kuid sageli ka jooksude grupeerimisel (Joonis 3). Ainult üks vastanutest väitis, et käsitsi ümber kirjutamise vajadust otseselt ei ole.



Joonis 3. Käsitsi andmete ümberkirjutamise vajadus jooksude haldamisel.

Viimase küsimusena uurisid töö autorid spordiürituste organiseerijatelt, kas tunnevad puudust või on huvitatud sellisest rakendusest, mis ühendab nii registreerimise, komplekteerimise, punktihalduse ja protokolli väljastamise funktsioonid ning kõikidelt vastanutelt saadi jaatav vastus.

Uurides täpsemalt veel erinevaid lahendusi, mida küll intervjuudes välja ei toodud, kuid mis on mujal maailmas küllaltki populaarsed platvormid, võib siinkohal mainida näiteks RaceEntry ning Eventbrite rakendusi. Need platvormid pakuvad samuti teiste seas selliseid funktsioone nagu registreerimine ja tulemuste sisestamine, kuid erinevalt autorite arendatud süsteemist, puudub neil oluline jooksude komplekteerimise funktsionaalsus. Küll aga leidub RaceEntryl erinevaid võistlustel kasutatavate taimerite liideseid, mis võimaldavad tänu kiibitatud jooksunumbritele tulemused automaatselt süsteemi saata. [4], [5]

Autorid analüüsisid, kas ja millistest loetletud rakendustest saab võtta üle praktikaid või meetodeid, mis võiksid arendatavas projektis edukalt toimida. Leiti, et Tartu KEVEK ja Tartu KALEV staadionijooksud kasutavad registreerimisankeeti, mida autorid saavad näitena kasutada oma registreerimisankeedi koostamisel. Samuti said autorid kasutada Sportity rakendusele sarnast automaatset protokolli genereerimise võimalust.

3 Metoodika

Antud peatükis kirjeldatakse Kohila Spordikeskuse varasemalt kasutatud etapil osalejate andmete haldamise protsessi, kasutatud tehnoloogiaid ja meetodeid ning autorite tööprotsessi.

3.1 Objekti kirjeldus

Enne projekti algust oli Kohila Spordikeskuse pikamaajooksu sarja registreerimisprotsess põhiliselt manuaalne ja kogu funktsionaalsus toimus MS Excel programmis. Osalejate registreerimine ja gruppidesse jagamine toimus käsitsi ning osalejad said registreerida läbi Kohila Spordikeskusele suunatud telefonikõne või e-kirja. Andmebaasina kasutati registreerunute kirjeid MS Excel programmis.

Olemasoleva protsessi puhul olid suurimateks murekohtadeks suur aja- ja ressursikulu ning suurem avatus inimlikele vigadele, näiteks võib registreerunute haldajal jääda mõne osaleja e-kiri kahe silma vahele või käsitsi andmeid Excelisse sisestades kogemata jooksjale vale eeldatav aeg. Osaleja jaoks puudus mugavus teha kõike samalt veebilehelt, sest etapi vaatamiseks oli vaja külastada Kohila Spordikeskuse või Eesti Kergejõustikuliidu kodulehte, registreerimiseks pidi avama meili või helistama. Samuti puudus võimalus kontrollida, kas registreering on olnud edukas ning tulemusi sai vaadata alles paar nädalat hiljem kergejõustiku või Kohila kodulehelt. Seega oli vaja lahendust, mis automatiseeriks protsessi, vähendaks vigade riski ja koondaks kõik kasutaja tegevused ühele lehele.

Lisaks registreerimisprotsessile oli vajadus efektiivsema jooksude komplekteerimise ja tulemuste väljastamise lahenduse järele. Mõlemad protsessid sisaldasid endas palju käsitsi andmete sisestamist või kirjutamist, mis suurendas samuti võimalikke vigu ja ajakulu.

Autorite poolt loodud süsteem võimaldab komplekteerida jooksud vastavalt jooksjate eeldatavale distantsi läbimise ajale ning neid soovi korral ümber kohandada. Süsteem võimaldab eksportida komplekteeritud jooksude nimekirja, mis kinnitatakse spordikeskuse uksele, kust võistlejad saavad näha oma jooksu ja rada. Lisaks hõlmab

rakendus tulemuste sisestamise funktsionaalsust, tulemuste vaatamist nii üldarvestuses kui ka vanuseklasside lõikes ja tulemuste eksportimist protokolli näol.

3.2 Kasutatud tehnoloogiad ja meetodid

Projekti teostamiseks kasutati erinevaid tehnoloogiaid ja programmeerimiskeeli. Rakendus loodi SaaS meetodil, kuna see võimaldab lihtsast ligipääsu nii kliendile kui ka osalejatele. SaaS ei nõua kliendilt rakenduse installimist, vaid brauserit ja võrguühendust [6]. Andmebaasi loomiseks kasutasid töö autorid MySQL andmebaasi, kuna see pakub laialdast funktsionaalsust, stabiilsust ning on ulatuslikult kasutatav avatud lähtekoodiga süsteem. Kasutajaliidese loomisel kasutati React.js raamistikku koos JavaScriptiga, kuna see võimaldab kiiret ja interaktiivset veebirakenduste arendamist ning pakub mitmekesist komponentide ja funktsionaalsuste valikut. Selleks, et luua võimalikult kasutajasõbralik kasutajaliides, kasutasid projekti koostajad kasutajaliidese disainimiseks Figma platvormi. Lisaks kasutati projektis GitLabi versioonihaldust, et meeskond saaks tõhusalt koostööd teha, koodi hallata ja jälgida arendusprotsessi kulgu. Aja logimise eesmärgil kasutati Toggl keskkonda.

3.3 Tööprotsessi kirjeldus

Tööjaotus meeskonnas oli järgnev:

- Henri: arhitektuuri loomine, andmebaasi loomine, peamine *back-end* loogika ja API päringud *back-end*'is, jooksude ja tulemuste eksportimine protokolli näol, jooksude automatiseeritud komplekteerimine, valideerimised, rakenduse kasutusele võtt (*deploy*), testid.
- Kadri-Liis: peamine *front-end* loogika ja API päringud *front-end*'is, sisse logimine ja autentimine, jooksude komplekteerimise manuaalsed funktsionaalsused (nt *drag-and-drop* ehk võimalus lohistada võistlejaid ühest positsioonist teise, rinnanumbrite sisestamine, rajanumbrid), disainide loomine ja implementeerimine, kasutusjuhend kliendile.

Täpsustuseks oluline märkus, et tööjaotus toodi välja peamise *back-end* ja *front-end* loogika põhjal. Tegelikkuses töötasid mõlemad liikmed nii *front-end* kui ka *back-end*

funktsionaalsuste juures, kuid vastavalt ülesannete sisule oli mõnel ülesandel suurem seos *front-end*'iga, teistel aga *back-end*'iga, mis toob kaasa selle erinevuse.

Projekti arendamiseks kasutas meeskond agiilset arendusmeetodit, mille eeliseks on selle paindlik ja iteratiivne lähenemine tarkvaraarendusele. Agiilse arenduse kasuks räägivad eelkõige selle väärtused. Esiteks rõhutab agiilne arendus tarkvaraarendajate ühtsust ja omavahelist suhet. Teiseks on oluline tarkvarameeskonna poolt toota testitav töötav tarkvara ning arendajatel soovitatakse hoida kood võimalikult lihtne, arusaadav ja tehniliselt võimalikult kõrgetasemeline, et vähendada dokumenteerimiskoormust. Kolmandaks tähtsaks agiilse arenduse komponendiks on arendajate ja klientide vahelised suhted ning koostöö. Agiilne arendus on keskendunud sellele, et projekti käivitamisel saaks klient koheselt oma ärile väärtust. Neljandaks peaks arendusgrupp oma tarkvaraarendajate ja klientide esindajatega olema hästi informeeritud, pädev ja arvestama arendusprotsessi jooksul tekkivate võimalike kohandamisvajaduste ja n-ö ümbermängimistega. Osalejad on valmis muudatusteks ja loodud kokkulepped toetavad neid täiendusi. [7]

Agiilse meetodina kasutati projektis Scrumi raamistikku. Scrumi lähenemine on välja töötatud süsteemide arendusprotsessi halduseks. Metoodika keskendub sellele, kuidas meeskonnaliikmed peaksid toimima, et arendada süsteem pidevalt muutuvas keskkonnas paindlikult. Scrumi põhimõte seisneb selles, et süsteemiarendus hõlmab mitmeid keskkonna- ja tehnilisi muutujaid, mis arenduse käigus tõenäoliselt muutuvad ning seetõttu muutub protsess ettearvamatuks ja keeruliseks. See nõuab omakorda süsteemide arendusprotsessi paindlikkust, et see saaks efektiivselt muutustele reageerida. [7]

Scrumi arendusfaas koosneb sprintidest ning käesolev arendusprotsess jagunes nelja kahenädalasse sprinti. Üle päeva andsid meeskonnaliikmed ülevaate, mis täpsemalt hetkel käsil on ja millised takistused esinevad. Iga kahe nädala tagant kohtuti ka kliendiga, et saada uut täiendavat sisendit süsteemi kohapealt, analüüsida tehtud tööd ning arutada võimalikke parandusi.

Oma ülesannete jälgimiseks ja projekti haldamiseks sobis hästi GitLab, võttes arvesse, et tiimiliikmetel oli juba varasemalt kokkupuude antud keskkonnaga. GitLabis sai loodud projekt ja projekti alla erinevad ülesanded (*issue*'d), millele oli võimalik lisada vastutaja (*Assignee*), staatus (nt *Developing, In Testing, Done*) ja silt (*label*), mis näitab, kas antud

ülesanne puudutab *front-end*'i, *back-end*'i või andmebaasi muudatusi. Antud keskkond oli vajalik, et meeskonnal oleks hea ettekujutus projekti hetkeseisust ning edasiliikumistest.

4 Töö tulemused

Käesolevas peatükis keskendutakse töö tehnilisele teostusele, mis hõlmab täpsemaid projekti nõudeid, kasutusjuhte, arhitektuuri ja disaini, koodi ning teste.

Autorid kaardistasid nõuded, võttes arvesse kliendi vajadusi ning osalejate ootusi, mis pandi kirja nii kliendi kui ka osalejatega otse suheldes.

4.1 Nõuded

Järgnevalt esitatakse autorite poolt kaardistatud üldised (mittefunktsionaalsed) nõuded, kliendi ning kasutaja nõuded.

4.1.1 Üldised nõuded

Kasutajaliides:

- Süsteem peab pakkuma kasutajasõbralikku kasutajaliidest, mis võimaldab kasutajatel mugavalt navigeerida ja süsteemi funktsioone kasutada.
- Kasutajaliides peab toimima sujuvalt erinevates seadmetes ja ekraanisuurustes.
- Kasutajaliidese disain peab olema sarnane kliendi kodulehe kujundusega.

4.1.2 Kliendi nõuded

Sisse logimine ja autentimine:

- Süsteem peab võimaldama kliendil sisse logida oma kontole, kasutades turvalist autentimisprotsessi läbi kasutajanime ja parooli sisestamise.
- Autentimisfunktsioon peab tagama kliendi privaatsuse ja andmete turvalisuse.

Komplekteerimine:

- Süsteem peab võimaldama automaatselt komplekteerida jooksjaid vastavalt nende eeldatavale jooksuajale ja grupi suurusele.
- Kliendil peab olema võimalus sisestada uus jooksja nii registreerunute kui ka komplekteerimise vaates.
- Kliendil peab olema võimalus sisestada võistlejate rinnanumbrid.
- Kliendil peab olema võimalus vajadusel manuaalselt muuta grupi koosseisu.

- Klient peab saama eksportida komplekteeritud jooksud dokumendi kujul.

Tulemuste haldamine:

- Süsteem peab võimaldama kliendil sisestada jooksjate tulemusi.
- Süsteem peab võimaldama kliendil tähistada võistlejad, kes ei tulnud kohale.
- Klient peab saama vaadata võistlejate tulemusi nii üldarvestuses kui ka vanuseklasside kaupa.
- Klient peab saama eksportida tulemused protokolli näol.

4.1.3 Kasutaja nõuded

Registreerimisfunktsionaalsus:

- Süsteem peab võimaldama jooksjatel end registreerida Kohila pikamaajooksu sarjale, esitades nõutud isikuandmed.
- Registreerimisprotsess peab olema selge ja lihtsasti jälgitav ning tagama, et kõik vajalikud andmed on täidetud.

Ülevaade etappidest:

- Kasutajal peab olema võimalus näha tulevasi etappe ja registreeruda neile.

Ülevaade registreerunutest:

- Kasutajal peab olema võimalus näha teisi registreerunuid ja kontrollida, et registreering läks läbi.

Ülevaade tulemustest:

- Kasutajal peab olema võimalus näha toimunud etapi tulemusi nii üldarvestuses kui ka vanuseklasside lõikes.

4.2 Kasutusjuhud

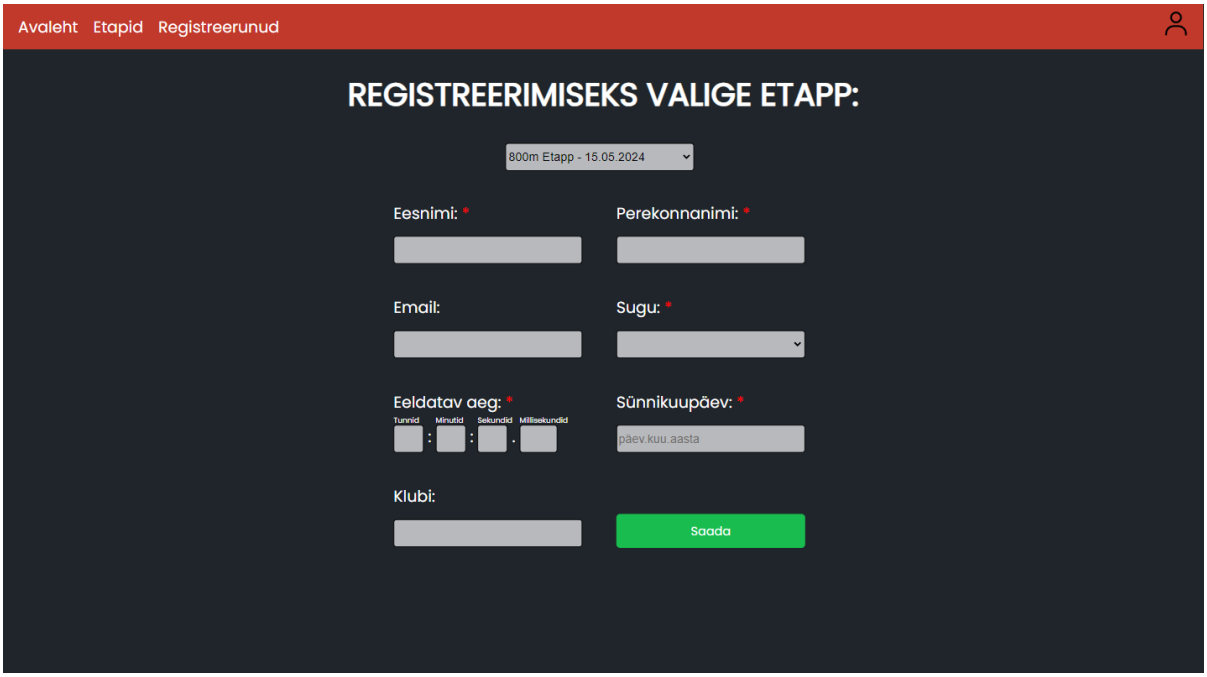
Järgnevalt on esitatud Kohila seeriajooksude süsteemi kasutusjuhud. Täpsustuseks tuleb mainida, et antud kontekstis mõeldakse kasutaja all etapil osalejat või registreerunut ning kliendi all administraatorit, kes pääseb ligi süsteemi kõikidele funktsioonidele.

Kasutusjuht: Registreerimine

Tegutseja: Kasutaja

Kirjeldus:

1. Kasutaja valib rippmenüüst etapi, millel soovib osaleda (Joonis 4).
2. Kasutaja täidab nõutud väljad.
3. Kasutaja vajutab nuppu „Saada“, mis edastab tema andmed Kohila seeriajooksude süsteemi.



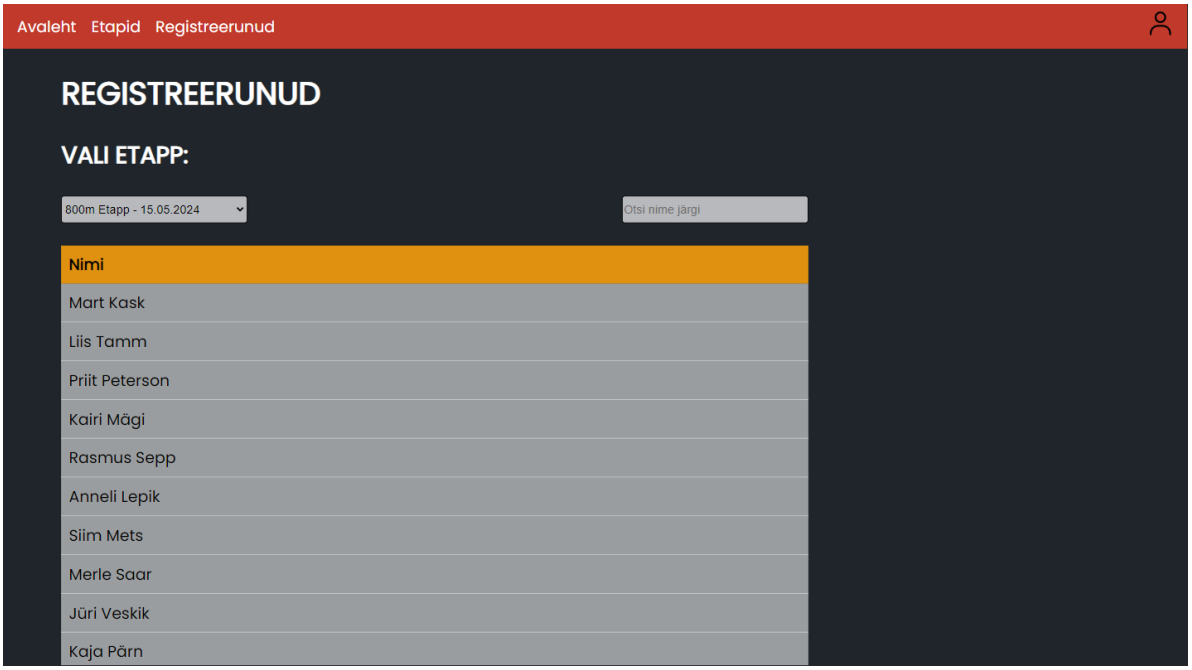
Joonis 4. Registreerimisvaade.

Kasutusjuht: Registreerunute vaatamine

Tegutseja: Kasutaja või klient

Kirjeldus:

1. Kasutaja vajutab navigeerimisribal lingile „Registreerunud“.
2. Kasutaja valib rippmenüüst soovitud etapi.
3. Kasutajale avaneb ülevaade valitud etapil osalejatest (Joonis 5).
4. Kasutajal on võimalus otsida võistlusel osalejat, kasutades otsinguriba tabelist üleval paremal.



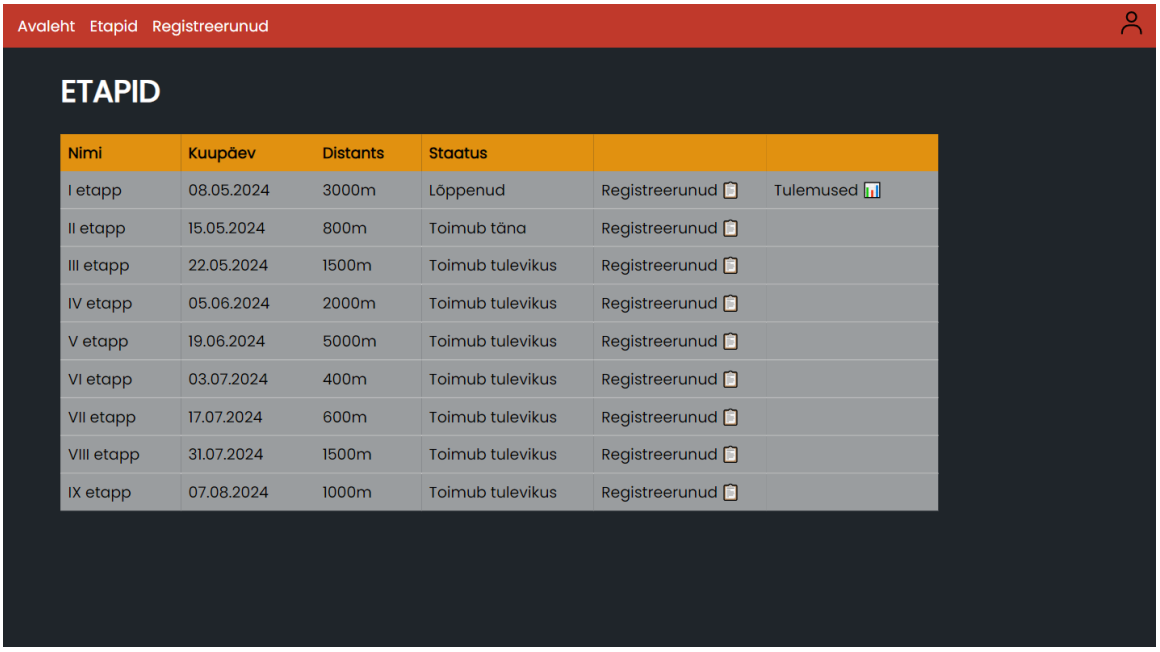
Joonis 5. Registreerunute vaade.

Kasutusjuht: Etappide vaatamine

Tegutseja: Kasutaja või klient

Kirjeldus:

1. Kasutaja vajutab navigeerimisribal lingile „Etapid“.
2. Kasutajale avaneb vaade lähenevatest etappidest (Joonis 6).
3. Kasutaja saab vaadata etapile registreerunuid, klikates vastava etapi juures lingile „Registreerunud“.
4. Kasutaja saab vaadata toimunud etapi tulemusi, klikates vastava etapi juures lingile „Tulemused“.



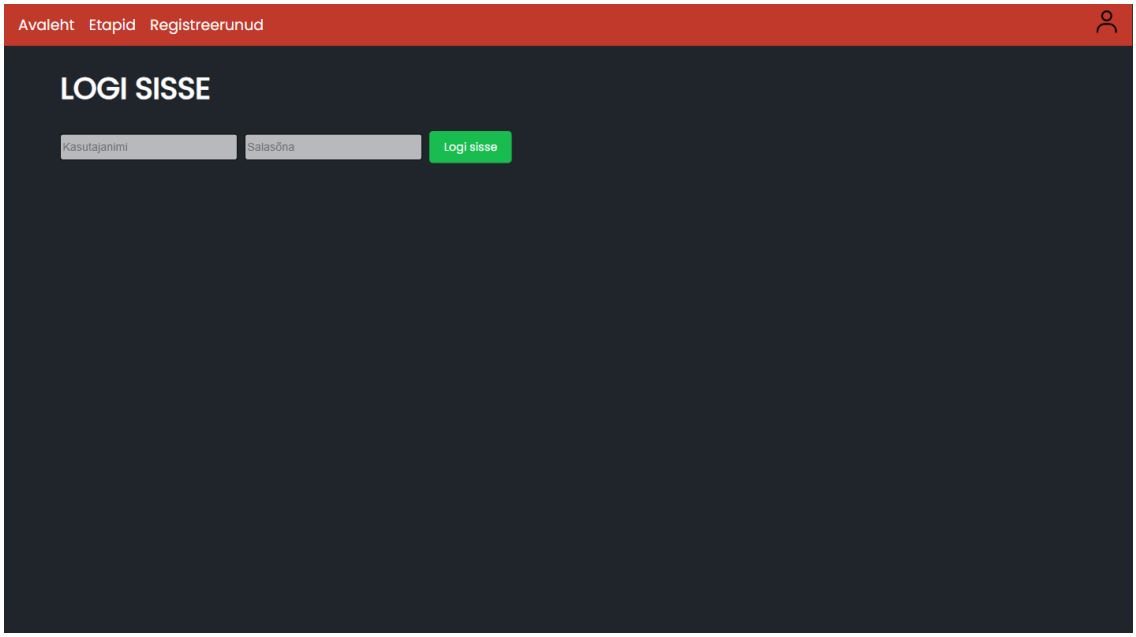
Joonis 6. Etappide vaade.

Kasutusjuht: Sisselogimine

Tegutseja: Klient

Kirjeldus:

1. Klient vajutab navigeerimisribal paremal paiknevale „logi sisse“ ikoonile.
2. Klient täidab vajalikud väljad.
3. Klient vajutab nuppu „Logi sisse“, mis õigete andmete sisestamisel teostab kliendi autentimise ja sisselogimise (Joonis 7).
4. Kliendil avaneb vaade välja logimise nupuga (Joonis 8).



Joonis 7. Sisselogimise vaade.

Kasutusjuht: Väljalogimine

Tegutseja: Klient

Kirjeldus:

1. Klient vajutab navigeerimisribal paremal paiknevale „logi sisse“ ikoonile.
2. Klient vajutab nupule „Logi välja“ (Joonis 8).

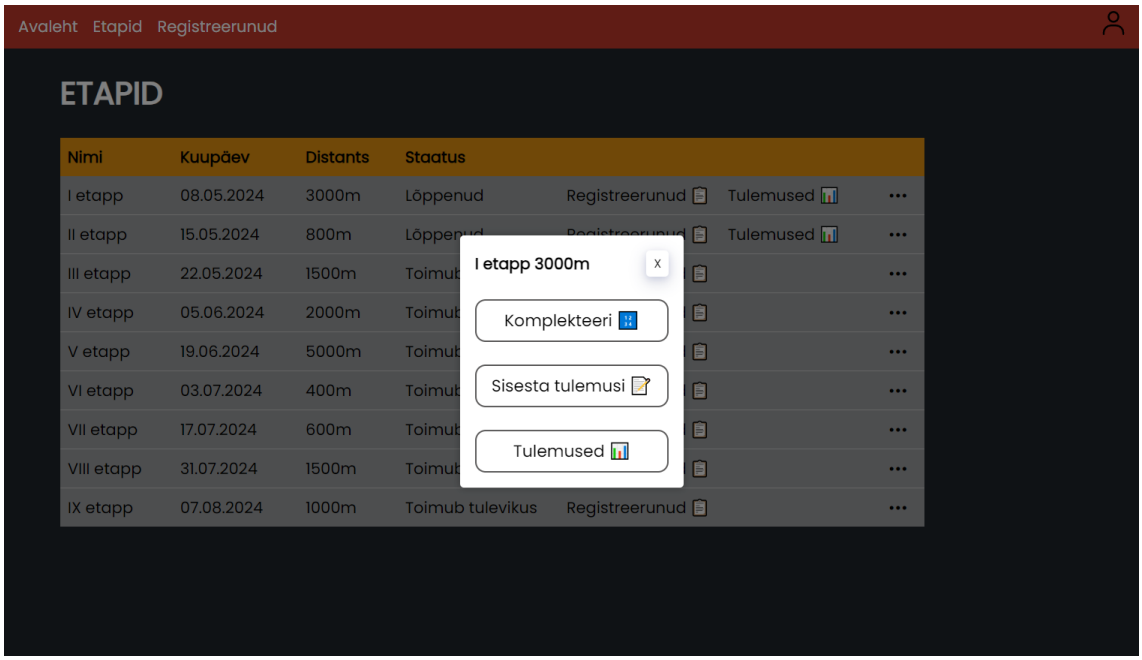
Joonis 8. Väljalogimise vaade.

Kasutusjuht: Muude toimingute vahel navigeerimine kliendi vaates

Tegutseja: Klient

Kirjeldus:

3. Klient vajutab navigeerimisribal lingile „Etapid“.
4. Klient vajutab soovitud etapi juures paremal ääres olevale kolmele täpile.
5. Klient saab vastavalt vajadusele navigeerida (Joonis 9) antud etapi komplekteerimisvaatesse (nupuga „Komplekteeri“) tulemuste sisestamise vaatesse (nupuga „Sisesta tulemusi“) ning tulemuste vaatesse (nupuga „Tulemused“).



Joonis 9. Muud toimingud kliendipoolses etappide vaates.

Kasutusjuht: Võistleja lisamine kliendi vaates

Tegutseja: Klient

Kirjeldus:

1. Klient vajutab navigeerimisribal lingile „Etapid“.
2. Klient vajutab tabelis soovitud etapile.
3. Kliendile avaneb vaade antud etapile registreerunutest (Joonis 10).
4. Klient vajutab nupul „Lisa võistleja“.
5. Klient täidab nõutud väljad (Joonis 11).
6. Klient vajutab nupul „Saada“.

Kasutusjuht: Võistleja eemaldamine

Tegutseja: Klient

Kirjeldus:

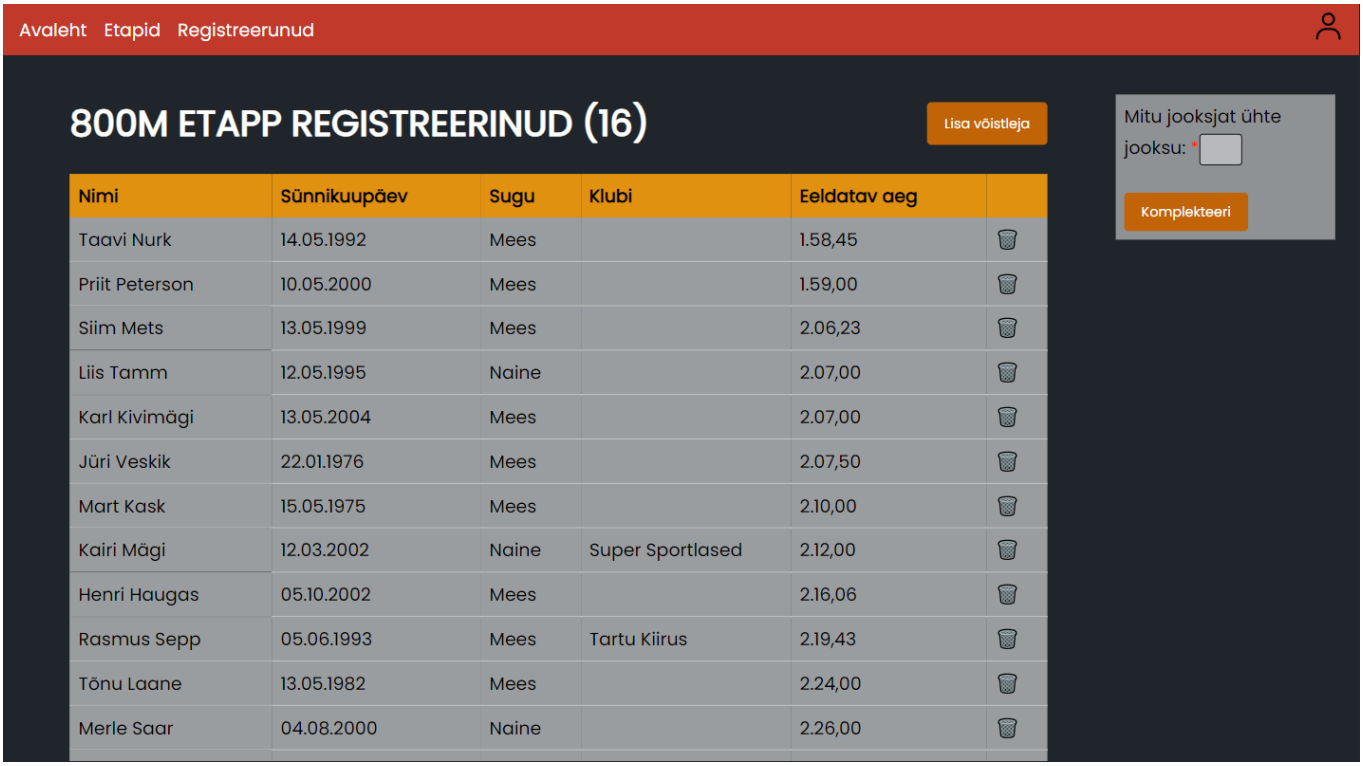
1. Klient vajutab registreerunute vaates (Joonis 10) prügikasti ikoonile vastava võistleja juures, keda soovib eemaldada.
2. Kliendile avaneb küsimus kinnitamiseks, kas soovib võistleja eemaldada.
3. Klient vajutab võistleja kustutamiseks nupule „Kustuta“.

Kasutusjuht: Komplekteerimine

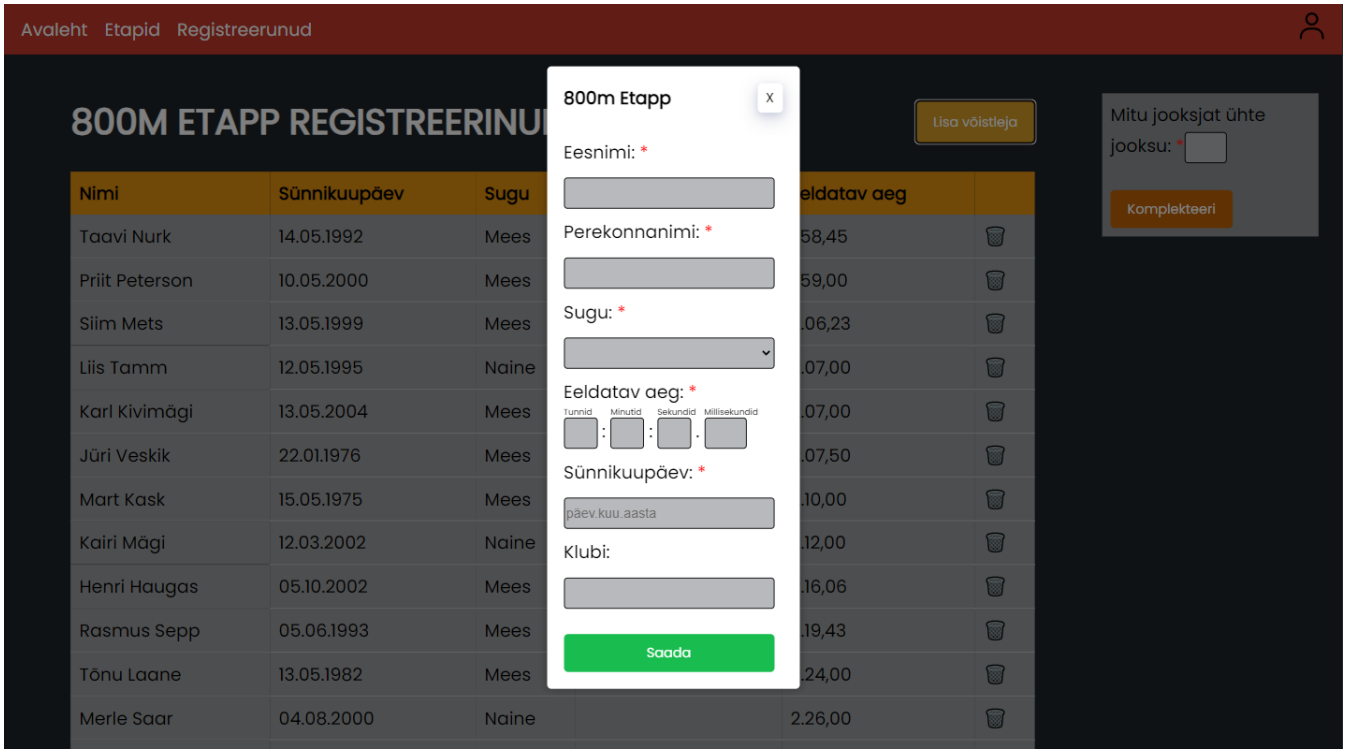
Tegutseja: Klient

Kirjeldus:

1. Klient sisestab registreerunute vaates paremal asetsevale väljale arvu, mitu jooksjat ühes grupis peaks olema (Joonis 10).
2. Klient vajutab nupul „Komplekteeri“.
3. Kliendile avaneb komplekteerimisvaade (Joonis 12).
4. Klient sisestab võistlejate juures toodud väljadele rinnanumbrid.



Joonis 10. Kliendipoolne registreerunute vaade.



Joonis 11. Võistleja lisamine registreerunute vaatest.

Kasutusjuht: Jooksu lisamine

Tegutseja: Klient

Kirjeldus:

1. Klient vajutab komplekteerimisvaates nupule „Lisa jooks“ (Joonis 12).
2. Komplekteerimisvaates lisandub jooks, kuhu on võimalik lisada jooksjaid nii teistest gruppidest lohistades kui ka „+“ nuppu kasutades jooksu juures.

Kasutusjuht: Jooksude eksportimine

Tegutseja: Klient

Kirjeldus:

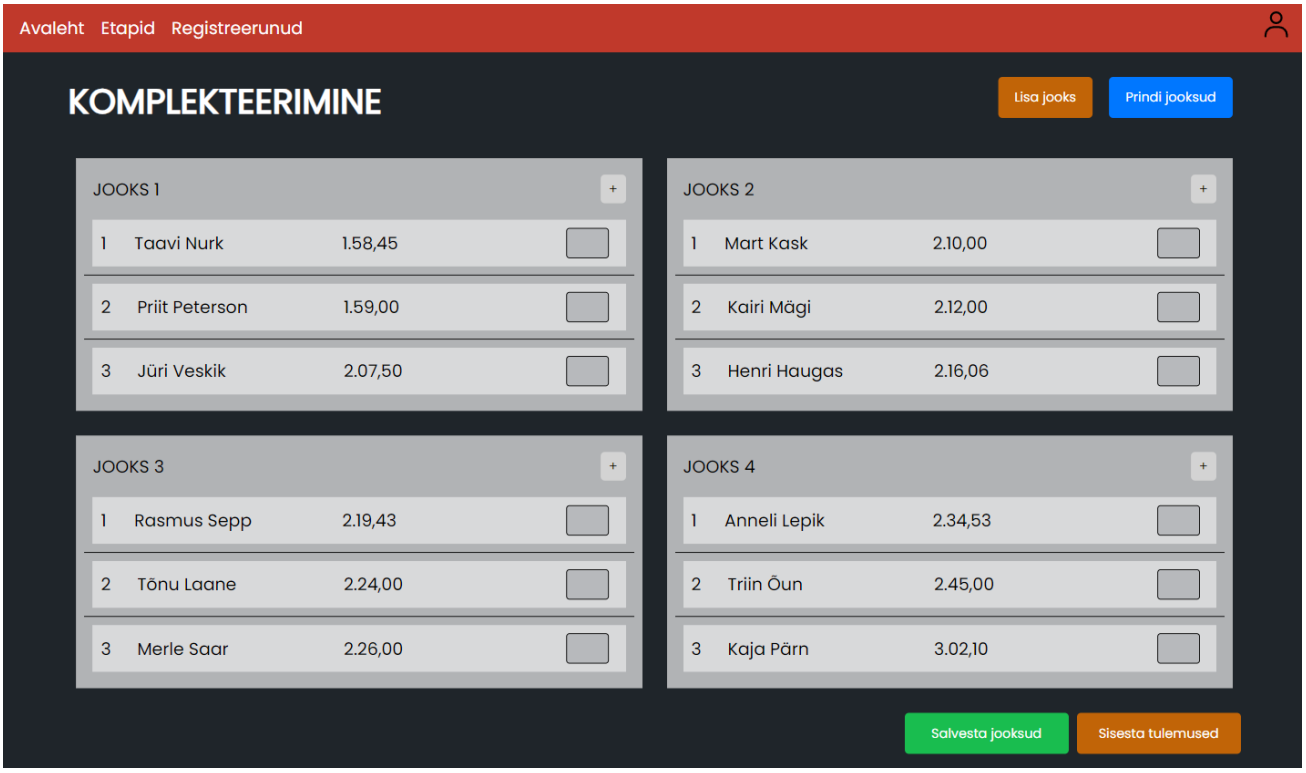
1. Klient vajutab komplekteerimisvaates nuppu „Salvesta jooksud“ (Joonis 12).
2. Klient vajutab komplekteerimisvaates nuppu „Prindi jooksud“.
3. Klient laeb faili alla (kausta „Downloads“ või „Allalaetud failid“).
4. Kliendil avaneb komplekteeritud jooksudega dokument (Lisa 5).

Kasutusjuht: Võistleja lisamine komplekteerimisvaates

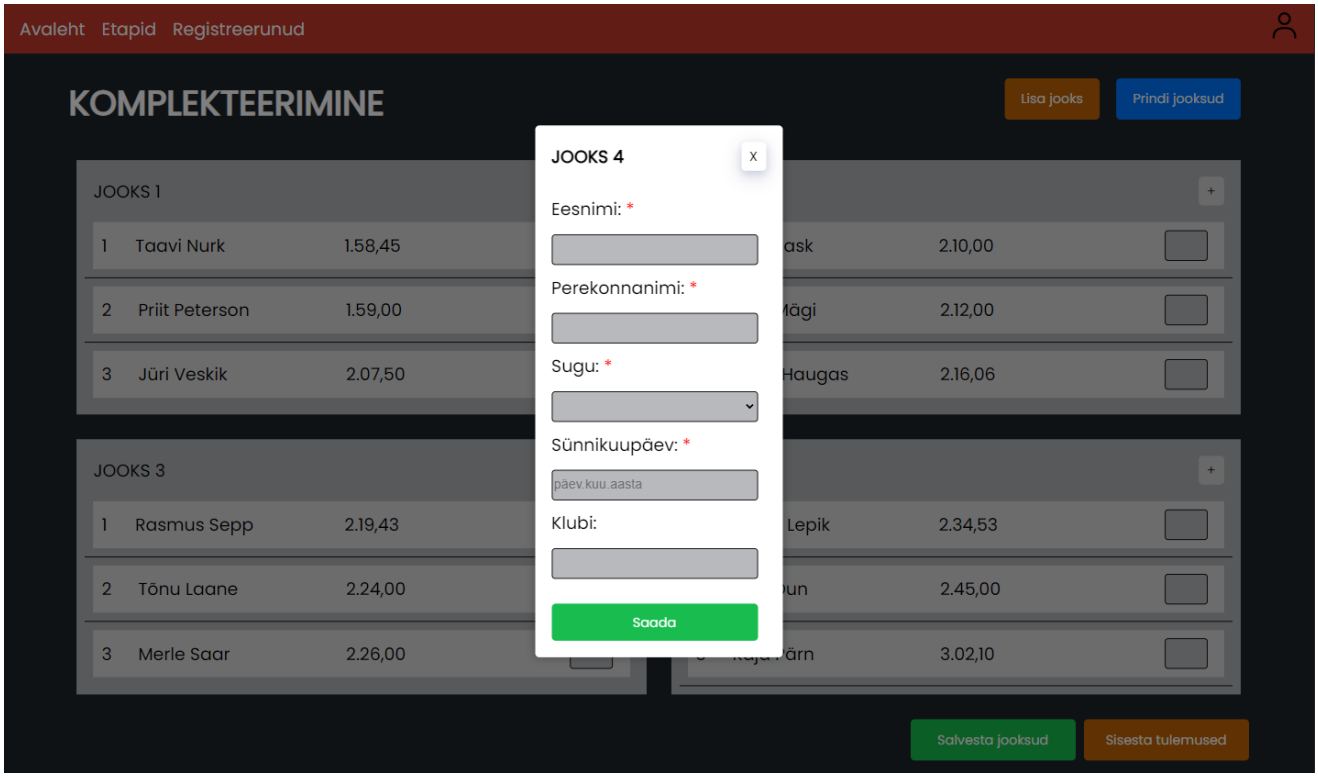
Tegutseja: Klient

Kirjeldus:

1. Klient valib komplekteerimisvaates jooksu, kuhu tahab jooksja lisada (Joonis 12).
2. Klient vajutab „+“ nuppu vastava jooksu juures üleval paremas nurgas.
3. Klient täidab nõutud väljad (Joonis 13).
4. Klient vajutab nuppu „Saada“.



Joonis 12. Komplekteerimise vaade.



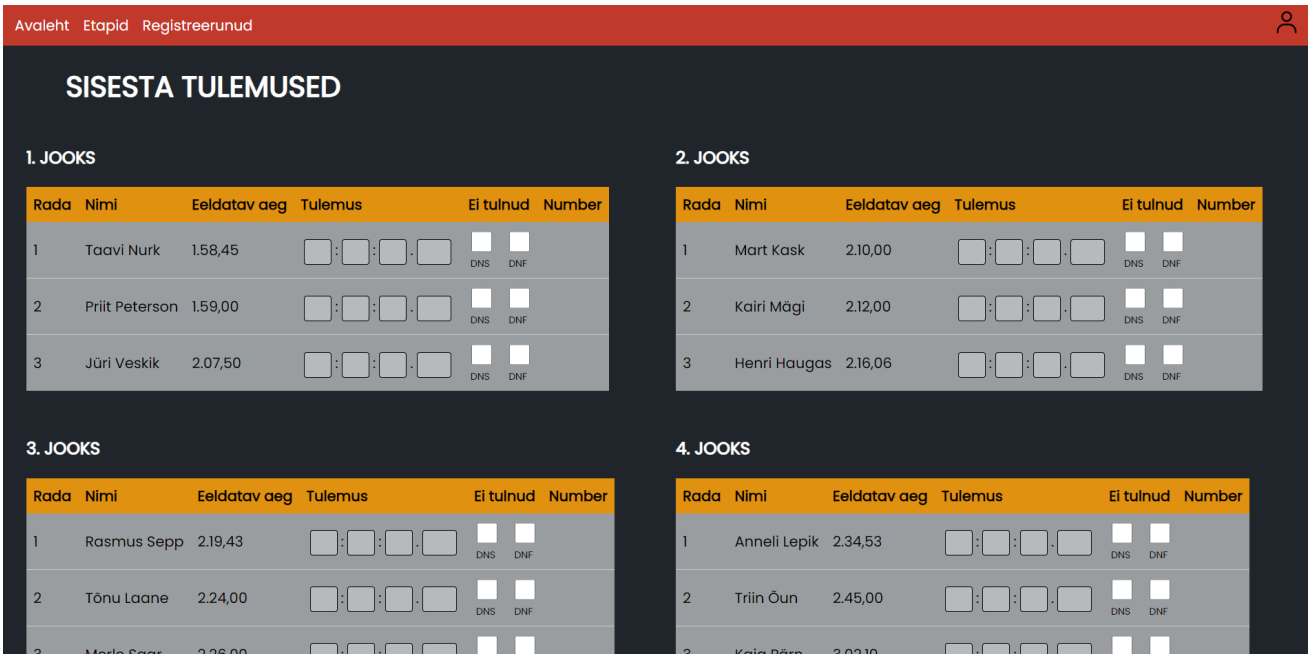
Joonis 13. Võistleja lisamine komplekteerimise vaates.

Kasutusjuht: Tulemuste sisestamine

Tegutseja: Klient

Kirjeldus:

1. Klient vajutab komplekteerimisvaates nupul „Sisesta tulemused“.
2. Kliendile avaneb tulemuste sisestamise vaade (Joonis 14).
3. Klient sisestab võistlejate tulemused.
4. Klient märgistab kastikese „DNS“ tulemuse kõrval, juhul, kui võistleja ei tulnud kohale.
5. Klient märgistab kastikese „DNF“ tulemuse kõrval, juhul, kui võistleja ei lõpetanud jooksu.
6. Klient vajutab nuppu „Salvesta tulemused“.



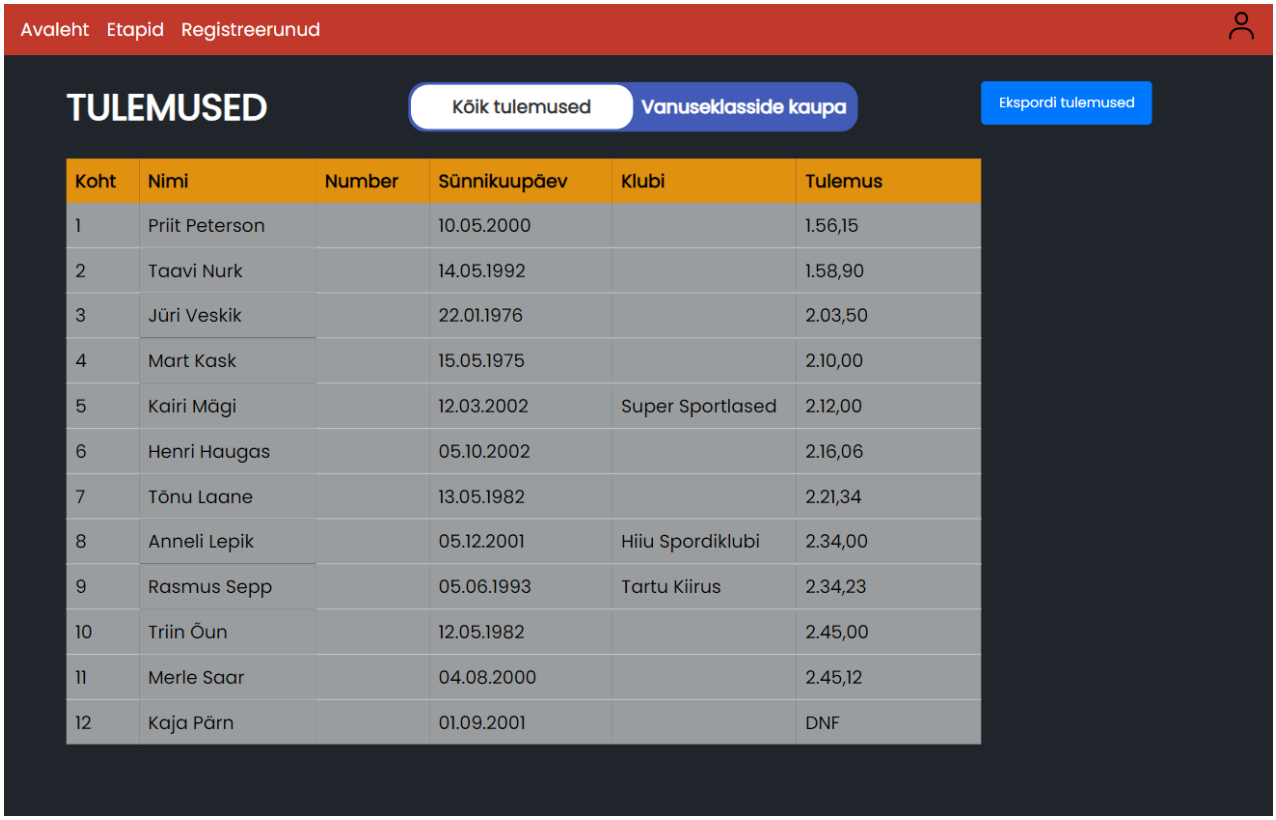
Joonis 14. Tulemuste sisestamise vaade.

Kasutusjuht: Tulemuste vaatamine

Tegutseja: Klient

Kirjeldus:

1. Klient vajutab tulemuste sisestamise vaates nupule „Tulemusi vaatama“.
2. Kliendile avaneb tulemused üldarvestuses (Joonis 15).
3. Valides ülemiselt *switch*-nupult „Vanuseklasside kaupa“, avaneb tulemuste vaade vanuseklasside lõikes (Joonis 16).



Koht	Nimi	Number	Sünnikuupäev	Klubi	Tulemus
1	Priit Peterson		10.05.2000		1.56,15
2	Taavi Nurk		14.05.1992		1.58,90
3	Jüri Veskik		22.01.1976		2.03,50
4	Mart Kask		15.05.1975		2.10,00
5	Kairi Mägi		12.03.2002	Super Sportlased	2.12,00
6	Henri Haugas		05.10.2002		2.16,06
7	Tõnu Laane		13.05.1982		2.21,34
8	Anneli Lepik		05.12.2001	Hiiu Spordiklubi	2.34,00
9	Rasmus Sepp		05.06.1993	Tartu Kiirus	2.34,23
10	Triin Õun		12.05.1982		2.45,00
11	Merle Saar		04.08.2000		2.45,12
12	Kaja Pärn		01.09.2001		DNF

Joonis 15. Tulemuste üldarvestus.

N 20-29	Koht	Nimi	Number	Sünnikuupäev	Klubi	Tulemus	Punktid
	1	Kairi Mägi		12.03.2002	Super Sportlased	2.12,00	12
	2	Anneli Lepik		05.12.2001	Hiiu Spordiklubi	2.34,00	10
	3	Merle Saar		04.08.2000		2.45,12	8
	4	Kaja Pärn		01.09.2001		3.02,00	

N 40-49	Koht	Nimi	Number	Sünnikuupäev	Klubi	Tulemus	Punktid
	1	Triin Õun		12.05.1982		2.45,00	12

M 20-29	Koht	Nimi	Number	Sünnikuupäev	Klubi	Tulemus	Punktid
	1	Priit Peterson		10.05.2000		1.56,15	12
	2	Henri Haugas		05.10.2002		2.16,06	10

M 30-39	Koht	Nimi	Number	Sünnikuupäev	Klubi	Tulemus	Punktid
	1	Taavi Nurk		14.05.1992		1.58,90	12
	2	Rasmus Sepp		05.06.1993	Tartu Kiirus	2.34,23	10

Joonis 16. Tulemused vanuseklasside kaupa.

Kasutusjuht: Tulemuste eksportimine

Tegutseja: Klient

Kirjeldus:

1. Klient vajutab tulemuste vaates nupule „Ekspordi tulemused“.
2. Klient laeb faili alla (kausta „Downloads“ või „Allalaetud failid“).
3. Kliendil avaneb tulemuste protokoll vanuseklasside lõikes (Lisa 6).

4.3 Arhitektuur ja disain

Peale kliendi nõuete ja soovide analüüsimist otsustasid autorid luua monoliitsel arhitektuuril põhineva rakenduse, lähtudes peamiselt välimiste teenustega suhtlemise vajaduse puudumisest. Monoliidi kasuks kaalus ka API päringute töötlemise kiirus, mille läbi viidud testidest on juttu peatükis 5.1.

Rakenduse *back-end* on kirjutatud JavaScriptis, kasutades Node.js raamistikku Express.js, mis võimaldab tõhusat HTTP-päringute töötlemist ning andmebaasiga suhtlemist. *Back-end* on mitmekihiline: see algab lõpp-punktist (*endpoint*), kus saabuv päring suunatakse vastava kontrolleri meetodile. Kontrolleri meetod sisaldab vajadusel valideerimist ning päringu töötlemise ja sobiva vastuse loogikat. Järgmise sammuna suunatakse päring edasi mudelikihile, kus toimub suhtlus andmebaasiga. Mudelikihis pannakse kokku andmebaasi päring, mis võib hõlmata andmete lugemist andmebaasist või uute andmete lisamist sinna (Lisa 2). Pärast andmebaasist lugemist või sinna kirjutamist koostatakse vastus, mis saadetakse tagasi kontrolleri kaudu lõpp-punkti ja seejärel kasutajale. Kui päring oli edukas, siis kuvatakse kasutajale vastav kinnitus. Kui aga päring ebaõnnestus, siis kuvatakse kasutajale veateade, mis sisaldab informatsiooni selle kohta, mis täpselt valesti läks.

Andmete salvestamiseks kasutavad autorid MySQL andmebaasi, mis on struktureeritud nelja tabeli abil, et säilitada erinevaid andmeid ja nende omavahelisi seoseid. Tabelid hõlmavad ürituste (*Events*) andmeid, võistlejate (*Registrations*) andmeid, tulemusi (*Results*) ning kasutajate (*Users*) autentimiseks vajalikke andmeid (Lisa 3).

Need tabelid on omavahel seotud võtmete abil, tagades andmete terviklikkuse ja konsistentsuse. Andmebaasi struktuur on kavandatud nii, et see toetaks rakenduse funktsionaalsust ja tagaks sujuva andmevahetuse rakenduse eri osade vahel.

Kasutajate tabelis hoitakse autentimisega seotud andmeid, nagu e-posti aadress ja parooli räsi, et võimaldada kliendil turvaliselt sisse logida ning võistlusi hallata.

Andmebaasi arhitektuuri disainides otsustasid autorid, kliendi soove arvestades, mitte kasutusele võtta kasutajapõhist sisselogimist. Sellest on lähemalt juttu peatükis 5.1. See tähendab, et kasutaja ei pea sisse logima, et ennast etapile registreerida. See on oluline arvestades etappidele registreerumiste ja tulemuste vahelist seost.

Iga registreerunu on unikaalne ja eristatav primaarvõtme *AthleteID* abil, lisaks on ta seotud vastava üritusega, millele ta registreerus, kasutades välisvõtit *EventID*. See tähendab, et registratsiooni andmetes sisalduv *EventID* viitab konkreetse ürituse identifikaatorile ürituste tabelis.

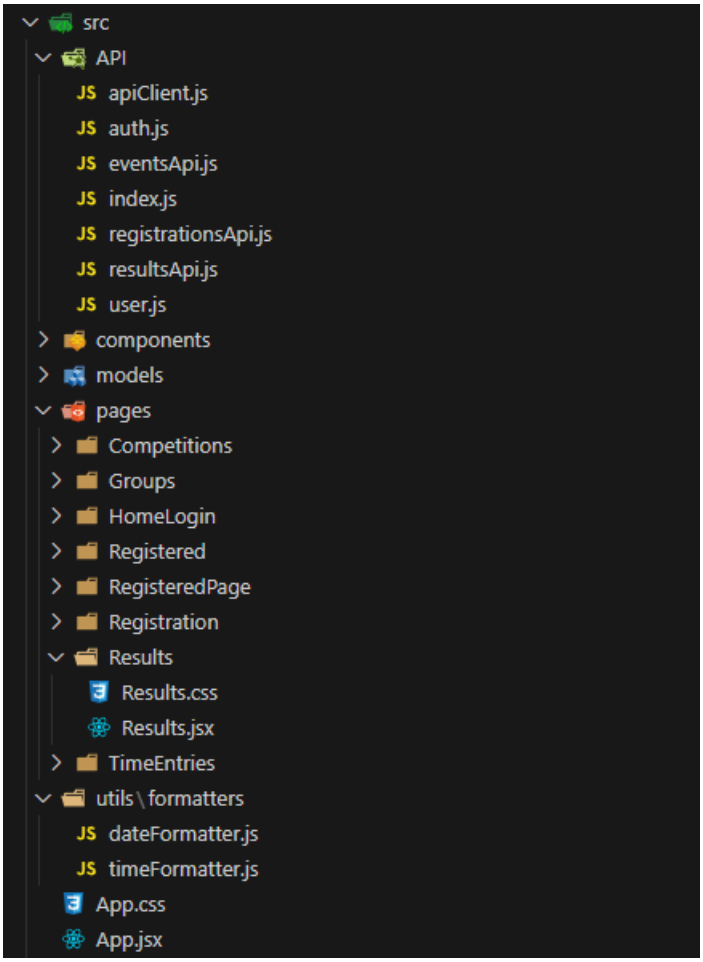
Ühel konkreetsel registreerumisel saab olla ainult üks tulemus. Seda tagatakse andmebaasi piirangu abil, mis määrab, et konkreetse ürituse jaoks saab iga võistleja (*AthleteID*) sisestada ainult ühe tulemuse (*ActualResult*), tagades seeläbi andmete unikaalsuse ja konsistentsuse. See aitab vältida topelt tulemusi või ebatäpseid andmeid tulemuste osas.

UI/UX disaini poole pealt lähtusid töö autorid kliendi nõudeid arvestades kliendi enda kodulehe disainist ning selle värvidest. Kuna see ei mänginud arenduse protsessis kõige prioriteetsemat rolli, siis ajaliste piirangute tõttu jäi see veidi rohkem skoobist kõrvale. Lisaks väljendas klient soovi oma kodulehe kujundus täielikult ümber teha ning sellega kaasneks lähiajal ka kogu käesoleva rakenduse disaini muutus.

Rakenduse *front-end* osas on failid ja komponendid paigutatud erinevatesse kaustadesse vastavalt nende kasutusele (Joonis 17). Kõik *back-end'*iga suhtlevad meetodid on „*API*“ kaustas, seal luuakse *backend*-ühendus, kasutades Axios HTTP klienti ning saadetakse päringuid andmete kuvamiseks või muutmiseks. „*components*“ kaustas asuvad taaskasutatavad komponendid, nagu näiteks nupud, hüpikakna moodulid ja navigeerimisriba, tänu millele vähendame koodiduplikatsiooni. *Back-end*'ist saabuvate JSON objektide kaardistamiseks (*mapping*) kasutame „*models*” kaustas defineeritud

klasse. Kuna vormindamist vajavaid andmeid on vähe, peamiselt ainult kuupäev ja jooksu tulemus ehk aeg, ning andmebaasi salvestatud kujul ei ole neid sobilik kasutajale kuvada, siis otsustasid autorid andmete vormindajat (*formatter*) kasutada vahetult enne kuvamist.

Vahelehtede haldamiseks rakendatakse struktuuri, kus iga erinev leht asub eraldi kaustas ning on nimetatud vastavalt selle kasutusele. Kõik vahelehtede kaustad kuuluvad omakorda „*pages*" kausta. Igal lehel on *.jsx* fail, milles asuvad HTML ja funktsionaalsuse loogika, ning *.css* fail, mis määrab visuaalse stiili ja kujunduse. Kujunduse osas lähtusid autorid www.kohilasport.ee kujundusest, et rakendus oleks kasutajale äratuntav ja panid rõhku võimalikult kasutajasõbraliku kasutajaliidese loomisele. Veebilehe kujundus on dünaamiline ja reageeriv (*responsive*), mis tähendab, et seda saab mugavalt kasutada nii arvutis, tahvelarvutis kui ka telefonis.



Joonis 17. Kasutajaliidese failide struktuur.

4.4 Kood

Koodi kirjutades jälgisid autorid tarkvaraarenduse põhimõtteid ning pöörasid erilist tähelepanu R. C. Martini "Clean Code" [8] teoses välja toodud põhimõtetele, milleks on koodi selgus, *DRY* (*Don't Repeat Yourself*) ehk ennast ei tohi korrata, koodi järjepidev refaktoorimine ja *SRP (Single Responsibility Principle)* ehk iga moodul või funktsioon peaks olema vastutav ainult ühe konkreetse funktsionaalsuse eest. Näiteks kasutatakse esiotsa rakenduses sünnikuupäeva valmiseks komponenti *DateOfBirthPicker* (Joonis 18), mida kasutatakse mitmes erinevas rakenduse osas. Tänu sellele väheneb koodiduplikatsioon ja tagatakse olukord, kus üks komponent vastutab ühe funktsionaalsuse eest.

```
const DateOfBirthPicker = ({ register, setValue, getValues, errors }) => {
  return (
    <div className="dob">
      <DatePicker
        id="DateOfBirth"
        {...register("DateOfBirth", {
          required: { value: true, message: "Sünnikuupäev on kohustuslik" }
        })}
        selected={getValues("DateOfBirth")}
        onChange={(date) => setValue("DateOfBirth", date, { shouldValidate:
        true })}
        dateFormat="dd.MM.yyyy"
        placeholderText="päev.kuu.aasta"
        peekNextMonth
        showMonthDropdown
        showYearDropdown
        dropdownMode="select"
        minDate={new Date(new Date().setFullYear(new Date().getFullYear() –
        150))}
        maxDate={new Date()}
        calendarStartDay={1}
      />
      <p className="error">{errors.DateOfBirth?.message}</p>
    </div>
  );
};
```

Joonis 18. Sünnikuupäeva valimise komponent.

Võistleja registreerimise puhul täidetakse *front-end'*is ankeet, kus valideeritakse kõik sisestatud väärtused ning tagatakse, et andmed sobituksid ettenähtud vahemikesse ja langeksid kokku andmebaasi väljade kriteeriumitega. Näiteks, et sekundid mahuksid

vahemikku 0 kuni 60. Vastasel juhul kuvatakse kasutajale täpne veateade, et kasutaja oskaks vigast sisendit muuta. Veatute andmete korral tehakse päring *back-end'*i lõpp-punkti pihta. Põhiliste meetoditena kasutame kõiki CRUD meetodeid (Joonis 19).

- CREATE (POST) – Andmete lisamiseks
- READ (GET) – Andmete pärimiseks
- UPDATE (PUT) – Andmete uuendamiseks
- DELETE – Andmete kustutamiseks

```
const router = express.Router();
router.get('/', registrationsController.getAllRegistrations);
router.get('/:id', registrationsController.getRegistrationById);
router.post('/', registrationsController.createRegistration);
router.put('/:id', registrationsController.updateRegistration);
router.delete('/:id', registrationsController.deleteRegistration);
```

Joonis 19. CRUD meetodite näited.

Seejärel suunatakse päring lõpp-punktist kontrollerisse. Kontrolleris kõigepealt valideeritakse muud andmed *regex*'iga, näiteks e-mail ning probleemi puhul tagastatakse kasutajale sõnum "Vale emaili struktuur, peab sisaldama "@", domeeni ja nime". Peale edukat valideerimist suunatakse andmed edasi mudelisse, kus toimub andmebaasi päringu käivitamine, kasutades *Structured Query Language* süntaksit.

Mudeli meetod on paigutatud kontrolleri meetodis *try-catch*'i sisse. Ehk juhul, kui andmebaas peaks vastama erroriga, püüab kontroller selle kinni ja tagastab kasutajale vastava staatuskoodiga veateate. Lisaks kasutavad rakenduse autorid koodis logimist (Joonis 20), juhuks kui serveri rikke tõttu peaksid andmed kaduma minema.

```
    createRegistration: async (registrationData) => {
        const { FirstName, LastName, Email, Sex, ExpectedResult, DateOfBirth,
RunningClub, EventID, RunNumber, LaneNumber, BibNumber } = registrationData;

        const [result] = await pool.query("INSERT INTO Registrations
(FirstName, LastName, Email, Sex, ExpectedResult, DateOfBirth, RunningClub,
EventID, RunNumber, LaneNumber, BibNumber) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?)", [FirstName, LastName, Email, Sex, ExpectedResult, DateOfBirth,
RunningClub, EventID, RunNumber, LaneNumber, BibNumber]);

        console.log(`${FirstName} ${LastName} registreerus ennast
${EventID}ndale etapile. Eeldatava ajaga ${ExpectedResult}`)

        return result;
    },
```

Joonis 20. Registreerimise päring andmebaasi.

Päringute autentimiseks kasutatakse *JWT*'d (*JSON Web Token*). Kui kasutaja on edukalt sisse logitud, genereeritakse talle *JWT*, mis antakse päringule kaasa küpsise kujul. Koodis rakendatakse *middleware* meetodit *verifyJwt*, mis kontrollib, kas saabunud päringul on küljes *token* (Joonis 21). Kui *token* puudub, tagastab meetod errori ja veakoodi.

```
const verifyJwt = async (req, res, next) => {
  const token = req.cookies.token;
  if (!token) {
    return res.status(401).json({ message: "Authentication token missing" });
  } else {
    try {
      const decoded = jwt.verify(token, jwtSecret);
      req.userId = decoded.id;
      next();
    } catch (err) {
      return res.status(403).json({ message: "Invalid token" });
    }
  }
};
```

Joonis 21. *Token*'i verifitseerimise meetod.

4.5 Testid

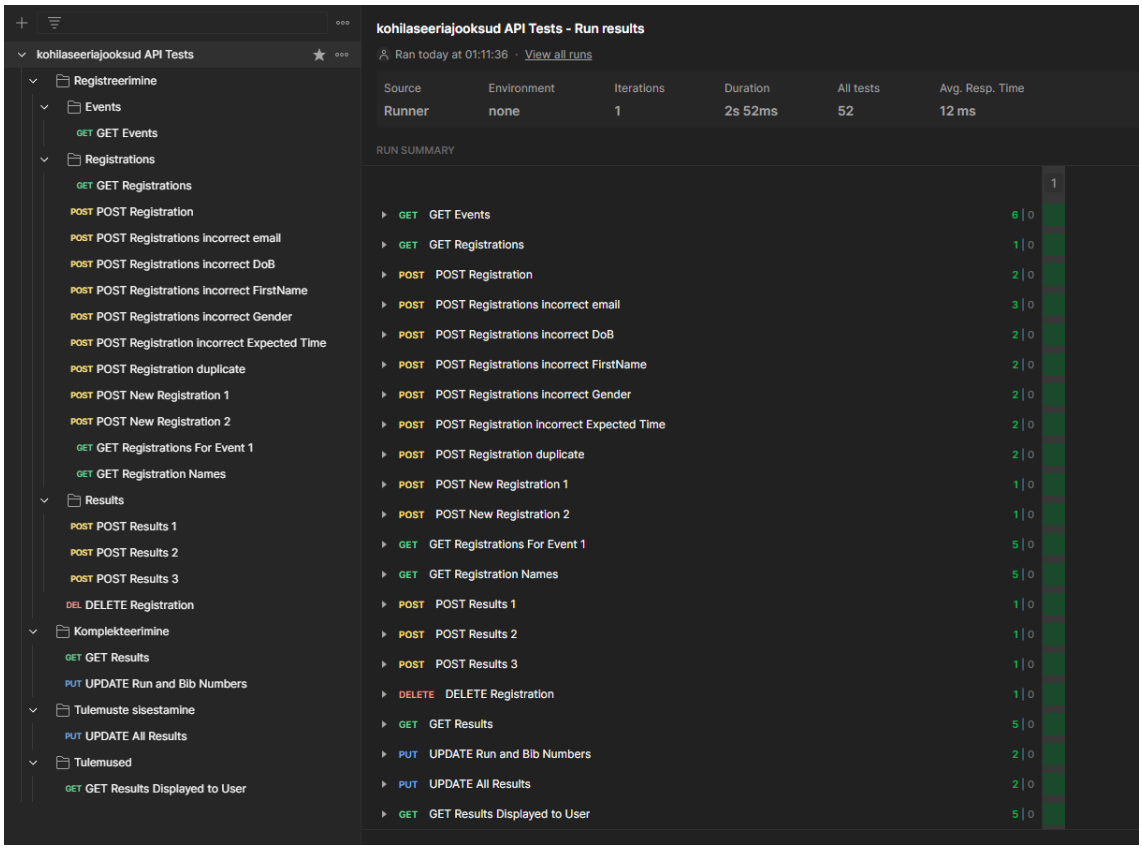
Testimine on oluline samm tarkvaraarenduse protsessis, kuna see võimaldab arendajatel kiiresti ja tõhusalt kontrollida nende loodud rakenduse programmi liidest. Postman pakub võimalust luua testimisskripte ning koheselt näha vastuseid ja vigu. Vigade avastamine

ja parandamine varases staadiumis säästab aega ja ressursse ning tagab rakenduse stabiilsuse ja usaldusväärsuse.

Autorid viisid arendamise käigus läbi hulgaliselt API teste. Testide põhiliseks eesmärgiks oli tagada kindlus, et kõik API päringud töötavad õigesti, ei võtaks vastu vales formaadis andmeid ja vale sisendi puhul annaks asjakohase veateate. Testimise käigus katsetati läbi kõik kasutuses olevad API päringud.

Testides (Joonis 22) on näha, et kogu etapi protsessi vältel kasutuses olevad *back-end* meetodid on testitud. Testimise hulka kuulusid järgmised kontrollid:

- Õige staatuskood
- Andmete valideerimisel veateate korrektsus
- Andmete õigsus pärast lisamise või muutmise toimingut
- Tagastatud objekti küsitud väljade olemasolu
- Väljade vormistuse korrektsus



Joonis 22. Postmani testid.

5 Analüüs ja järeldused

Käesolevas peatükis põhjendavad töö autorid tehnilist teostust, valideerivad tulemuse ning kirjeldavad projekti edasiarendust. Lisaks tuuakse välja teostatud tööde logid, autorite eneseanalüüsid ning meeskondlik hinnang.

5.1 Tehnilise teostuse põhjendus

Peatükis 4.1 toodud nõuded said koostatud läbirääkimistel kliendiga, kaardistades nende vajadusi seoses registreerimissüsteemiga. Autorid pidasid silmas kliendi prioriteete ning soovisid luua kliendile võimalikult mugava ja kasutajasõbraliku lahenduse. Algselt avaldas klient soovi rakendusele, mis võimaldaks osalejatel end etappidele registreeruda ning võimalusel osalejad gruppidesse komplekteerida, kuid lähemal analüüsimisel selgus, et oleks vajadus ka tulemuste sisestamise, protokolli eksportimise ja mitmete teiste väiksemate funktsioonide järele, näiteks rinnanumbrite sisestamine võistlejatele jne.

Autorid arendasid monoliitse arhitektuuriga rakenduse peamiselt selle lihtsuse tõttu, kuna kogu rakendus on ühes koodibaasis ja ei kasutata väliseid teenuseid. Kui monoliit on väike ja sellel on vaid mõned funktsioonid, on selle tugevateks külgedeks justnimelt arendamise, testimise, juurutamise ja skaleerimise lihtsus [9]. Lisaks on ühe rakenduse kasutusele võtmine (*deploy*) lihtsam kui mitme teenuse paigaldamine. Mikroteenuste [10] arhitektuur oleks tõhusam, kui rakendus peab käsitlema suuremat arvu päringuid. Selle eelisteks oleksid lihtne hooldus ja töökindlus, kuna mikroteenuse rike mõjutab ainult seda üht mikroteenusega seotud komponenti ning mitte teisi. Küll aga käesolev rakendus ei pea läbi töötama eriti palju päringuid ja ei kasuta väliseid mikroteenuseid, nii et töö autorid otsustasid teenuste jaoks kasutada üht andmebaasi ning seega kasutada monoliitset arhitektuuri.

K. Gos ja W. Zabierowski toovad monoliitse ja mikroteenuste võrdluse üheksandas peatükis [10] välja läbi viidud testid, kus sama masina peal võrreldi erinevate arhitektuuride HTTP GET ja POST päringute töötlemise kiirust. Esimeses testis tehti 30 000 GET päringut. Tulemustes selgus, et monoliit suudab käsitleda ca. 789 päringut sekundis, mikroteenuste jõudlus oli väiksem, nimelt parimal katsel ca. 600 päringut. Teise katse puhul (300 000 päringut) oli tulemus vastupidine. Siin on parimaks valikuks mikroteenuste arhitektuur, mis saavutas tulemuse 215 päringut sekundis, samal ajal kui

monoliitrakendus töötles ainult 180. POST päringute puhul olid tulemused analoogsed. Kuna lõputöö autorite rakendus ei vaja suure hulga päringute töötlemist, sobib monoliitne arhitektuur suurepäraselt.

Disaini aluseks võeti mitmeid tuntud tarkvaraarenduse mustreid, mis aitavad tagada süsteemi terviklikkust ja laiendatavust. SPAd kasutati seetõttu, et see võimaldab veebirakendusel dünaamiliselt värskendada ainult muutunud veebilehe osi ja vähendab sellega serveri koormust. Lisaks vähendab see kasutajaliidese reageerimisaega. SPA [11] puhul salvestatakse lähtekood ühe laadimisega ühelehelistes rakendustes. Protsessi ühelgi hetkel ei laeta lehekülge uuesti. Ühelehelistest veebirakendustes on mudelis mitmeid eraldiseisvaid üksikuid komponente ning terve lehe laadimise asemel värskendatakse komponente iseseisvalt iga kasutajatoimingu korral.

MVC disainimustrit rakendati seetõttu, et kood oleks selgem ning tulevikus lihtsam muudatusi läbi viia. MVC mustri põhiidee [12] on eraldada kasutajaliidesed nende aluseks olevatest andmetest. MVCs kuvab vaade (*View*) kasutajale teavet ja koos kasutaja interaktsiooni töötleva kontrolleriga (*Controller*) sisaldab veebirakenduse kasutajaliidest. Mudel (*Model*) on rakenduse osa, mis hõlmab endas nii vaates kuvatavat teavet kui ka loogikat, mis muudab antud teavet vastuseks kasutaja interaktsioonile.

Back-end'i disainides järgiti RESTful arhitektuuri põhimõtteid, kuna see võimaldab lihtsat suhtlust kliendi ja serveri vahel ning tagab API tõhusa kasutamise. RESTful APId koosnevad lõpp-punktidest ja igaüks neist täidab konkreetset funktsionaalsust. REST stiili pakkus välja Roy Fielding 2000. aastal oma doktorikraadis, tuues välja selle eesmärkidena jõudluse, skaleeritavuse, lihtsuse, muudetavuse ja töökindluse parandamise. Selleks, et antud eesmärke saavutada, leidsid HTTP-verbidest kasutust just DELETE, GET, POST ja PUT. [13]

Arenduses järgiti R. C. Martini „Clean Code" teosest pärinevaid printsiipe, et hõlbustada koodi loetavust, tiimitööd ja koodi skaleeritavust, kuna puhas kood koos loogiliste ja selgete nimetustega, muudab koodist aru saamise teisele inimesele tunduvalt lihtsamaks. Mida loetavam kood, seda kergem on läbi viia muudatusi ning samuti aru saada, kust võivad tekkida vead.

Kuna loodud rakenduse näol on tegemist lahendusega, mille käigus on API päringute järjekord oluline, sarnaselt Petstore'iga, kus ei saa näiteks enne looma tellida, kui selle

staatus on saadavaks muudetud, mida kirjeldavad Yi Liu ja teised autorid oma töös [14]. Sellest lähtuvalt võtsid autorid testide läbiviimiseks kasutusele RESTler tüüpi metoodika, mis keskendub API päringute jada õigsusele, päringute isoleeritud testimisele, API kutsete järjestuse laiendamisele ning heuristilise lähenemise kasutamisele uute kutsete lisamisel, tagamaks tervikliku ja usaldusväärse rakenduse toimimise. Testide puhul on kaetud kõik päringud, mis on kasutuses ühe etapi läbiviimise käigus alates registreerimisese valideerimisest, kuni kontrollini, et lõpuks näidatud tulemuste andmed on tõesed.

5.2 Kasutusjuhtude põhjendus

Registreerimisvormi koostamisel lähtusid autorid kõige tavapärasemast ankeedist, kus saab rippmenüüst valida etapi ning täita valideerimistega väljad. Näitena kasutati Tartu KEVEK ja Tartu KALEV staadionijooksude registreerimisankeeti.

Lähenevad etapid on toodud lihtsasti tabeli näol, kus saab iga etapi juures näha registreerunuid ning tulemusi, kui need on väljastatud. Sisse logituna lisandub kliendile iga etapi juures muude toimingute funktsioon, kust saab navigeerida nii antud etapi komplekteerimise, tulemuste sisestamise kui ka tulemuste vaatesse. Antud lingid teevad kasutaja ja kliendi jaoks navigeerimise lihtsamaks ning kiiremaks.

Etapile registreerunuid saab näha samuti tabeli kujul, valides rippmenüüst soovitud etapi või vajutades etappide vaates lingile „Registreerunud“. Tabel annab hea ülevaate osalejatest ning lisaks saab kasutada otsinguriba, et osalejal oleks mugav otsida kas enda, mõne tuttava või konkurendi nime.

Sisse logituna avaneb kliendile veidi teistsugune registreerunute vaade, kus lisandub võimalus osalejaid kustutada, kui nad on varasemalt teatanud, et pole siiski etapil osalemas. Lisaks saab seal vaates sisestada jooksjate arvu ühes grupis, mida võetakse arvesse jooksugruppide komplekteerimisel.

Komplekteerimisvaates on näha gruppe, mis on koostatud nende eeldatava distantsi läbimisaja alusel, kuna see on kergejõustikus tava, et püsiks tihe konkurents ja kõigil oleks võimalus maksimaalsele pingutusele. Osalejaid saab lohistamisfunktsiooniga jooksudes ümber kohandada, juhuks kui mõni osaleja soovib gruppi vahetada ning sageli lisatakse viimasesse jooksu tunduvalt rohkem osalejaid, kui algselt gruppidesse on

määratud. Samuti saab antud vaates lisada rinnanumbrid, kuna korraldajal on lihtne eristada rinnanumbreid vastavalt gruppidele ning seejärel neid osalejatele anda.

Nii kliendipoolses registreerunute kui ka komplekteerimisvaates on oluline, et klient saaks lisada uue osaleja, kuna on mõningaid sportlasi, kes pole end eelregistreerinud ning teevad seda kohapeal. Samuti on Kohila rõhutanud, et soovivad anda kõikidele võimaluse joosta, isegi kui nad jõuavad end registreerima pea viimasel hetkel. Osaleja lisamine käib läbi hüpikakna, et vältida uuele vahelehele navigeerimist.

Komplekteeritud jooksud saab eksportida pärast jooksude salvestamist, mis salvestab lisaks jooksude koosseisudele ka jooksjate rinnanumbrid. See on vajalik, et korraldaja saaks jooksud välja printida ning spordikeskuse uksele kinnitada. Iga osaleja saab sealt leida oma jooksu ja rinnanumbri.

Tulemused tuleb sisestada käsitsi, kuna hetkel ei ole Kohila Spordikeskusel integreeritud liidest, mis tulemused automaatselt süsteemi saadaks. Lisatud on ka märkeruudud, et märkida osalejad, kes ei tulnud stardiks kohale („DNS“ ehk *did not start*) ning osalejad, kes ei lõpetanud jooksu („DNF“ ehk *did not finish*). Nendel juhtudel tehakse vastav märge protokolli ning tulemus ei lähe kirja. Seega on antud märkeruutudel oluline roll protokolli genereerimisel.

Tulemusi saab vaadata nii üldarvestuses kui ka vanuseklasside lõikes, kuna see on olulise tähtsusega osaleja jaoks, et näha oma positsiooni mõlemas arvestuses. Lisaks annab vanuseklassiline arvestus aimu, milliseks kujuneb võistluse protokoll, mis samuti genereeritakse vanuseklasside alusel. Taaskord saab kõige mugavamalt ülevaate just tabeli kujul.

Tulemuste eksportimine on vajalik, sest protokoll on vaja esitada Eesti Kergejõustiku liidule, et nad saaksid sisestada tulemused oma andmebaasi.

5.3 Tulemuse valideerimine

Rakenduse tulemuse valideerimiseks saab analüüsida kahte juba toimunud etapi kulgemist, võrdlust teiste sarnaste rakendustega ning kliendi, kasutaja ja autorite tagasisidet.

Klient oli töö tulemusega väga rahul, sest enam ei pea aega kulutama eelregistreerunud sportlaste andmete ümberkirjutamisele, jooksude komplekteerimine on lihtsam ja jooksude dokument ning tulemuste protokoll genereeritakse automaatselt. Lisaks on kogu protsessis vähem võimalusi inimlikeks vigadeks. Peale esimest etappi tuli ka kliendilt paar uut vajadust, mille peale ei osanud kumbki osapool varem mõelda. Näiteks rinnanumbrite lisamine jooksude dokumendile, võimalus jooksjat kustutada ning kasutajasõbralikum navigeerimine kliendi vaadete vahel. Tänu "Clean Code" põhimõtete järgmisele ja heale arhitektuurile oli autorite muudatusi lihtne sisse viia ning järgmiseks etapiks olid kõik soovitud lisafunktsionaalsused olemas. Kliendilt saadud positiivset tagasisidet ei saa muidugi võtta määratleva valideeriva asjaoluna, sest arvestades nende lähtepunkti, oli antud tagasiside ootuspärane. Seetõttu otsustasid töö autorid kasutada valideerimiseks ka võrdlust teiste sarnaste lahendustega.

Võrdluses teiste lahendustega on autorite poolt loodud rakenduse eeliseks kõik-ühes lahendus, uudne komplekteerimise võimalus ning manuaalse töö ja protokolli eksportimise automatiseeritus. Ainukeseks manuaalseks toiminguks on tulemuste sisestamine. Samamoodi toimib see ka Sportityl, millel on kõige sarnasemad funktsionaalsused loodud rakendusega. Selle miinuseks on aga märkimisväärselt kõrge hind, mis ei mahu üldjuhul korraldajate eelarvesse. Sportity on hea suure ürituste korraldamiseks, kuid väiksemate ürituste puhul ei kompenseeriks tööjõukulude kokkuhoid Sportity kasutamist.

Kuna tegemist oli uudse võimalusega registreerimiseks ning polnud kindel, kuidas kasutajad sellele reageerida võivad, otsustasid autorid kooskõlas kliendiga alles jätta ka e-maili ja telefoni teel registreerimisvõimaluse. Vaatamata sellele, et erinevaid võimalusi eelregistreerumiseks oli kolm, osutus kõige populaarsemaks registreerimisvormi täitmine. Esimesel etapil osales 51 inimest, kellest 44 registreerus läbi vormi, 3 e-maili teel ning ülejäänud 4 registreerusid ennast füüsiliselt koha peal, tund aega enne starti. Nendest tulemustest võib järeldada, et kasutajate jaoks oli registreerimisvorm kõige mugavam viis. Osalejate seas tekitas meelerahu võimalus näha, et registreering on olnud edukas ja nimi on listis, kuid tundsid puudust võimalusest ennast nime järgi otsida. Ka see lisafunktsionaalsus viidi järgmiseks etapiks sisse.

Autorid on arvamusel, et üldjoontes on olnud töö edukas ning kõik seatud eesmärgid said täidetud, kuid kindlasti on ka palju ruumi edasiarendusteks, eelkõige kasutajamugavuse parandamiseks.

5.4 Äriline kasu

Kuna etapid toimuvad õhtuti, väljaspool standardset tööaega, pidid üritusega seotud töötajad mitu tundi varem kohale tulema ja alustama saabunud registreerunute meilidelt andmete ümberkirjutamist ja manuaalset jooksude komplekteerimist. Seega vähendab meie rakendus ajakulu, mis omakorda vähendab tööjõukulu.

Osalejad saavad ennast mugavamalt registreerida, vaadata etappe ning tulemusi, kõike samalt veebilehelt, mis suurendab nende rahulolu, seotust üritusega ja muudab järgmisele etapile tuleku rohkem tõenäolisemaks. See suurendab tulusid osalustasudest.

Paremat ülevaadet andmetest ja osalejate arvu saab kasutada erinevate analüüside tegemiseks, näiteks osalejate profiilide koostamiseks, mis aitab tulevikus paremini planeerida ja sihtida turundustegevusi.

5.5 Edasiarendus

Lähtudes peale jooksmise ka teistest valdkondadest ning vaadates suuremat pilti, võiks kindlasti väita, et selline komplekteerimise funktsionaalsus tuleks kasuks ka mujal. Oletades, et tekib olukord, kus on vaja moodustada suvalised tiimid kindla indikaatori alusel, saaks kasutada ära antud rakendust, muutes vaid paari grupeerimise protsessi osa. Vahetades eeldatava distantsi läbimise aja mõne teise kriteeriumiga, saab rakendusega täita juba veidi teistsugust eesmärki. Eeldatava aja asemel võib kasutada küsimusi, mis määraksid ära osaleja tugevused ning seejärel koostaks algoritm meeskonnad, kus kõikidel liikmetel on vastavateks rollideks vajalikud omadused. Näitena võib siin tuua autodega orienteerumise: küsimuste abil saab paika panna tiimi, kus on hea kaardilugeja, autojuht, laia silmaringiga või orienteerumispaika hästi tundev liige jne. Sarnaselt saavad sõbrad kokku tulles luua jalgpallimeeskonnad, mis oleksid võimalikult võrdsed, arvestades mängijate oskusi ja taset.

Rakendusi, mis võimaldavad grupeerimist teatud kriteeriumi alusel, leidub mitmeid, kuid üldjuhul toimub komplekteerimine nende puhul soo või taseme (skaalal 1-5) alusel, mis

kujutab endast pigem piiratud valikut. Sellist grupeerimist võimaldab näiteks Keamk keskkond [15].

Üks levinumaid kindlate kriteeriumite alusel meeskonna loomise algoritme on Gale-Shapley. Antud algoritmi käsitlevad oma artiklis Daniel Meulbroek, Daniel Ferguson, Mathew Ohland ja Frederick Berry täpsemalt. Nad toovad välja, et süstemaatiliste protsesside kasutamine ning õpilase informatsiooni arvesse võtmine meeskondade loomisel suurendab tõenäoliselt õpilase pealehakkamist projekti suhtes. Artiklis analüüsitud CATME Team-Maker algoritm sisaldab 20 sisseehitatud küsimust, mille seast tiimide koostaja saab valida, mis on olulised kriteeriumid tiimide grupeerimisel. Küll aga on seal puudusi küsimuste näol, mis võtaksid arvesse õpilase enda soove projektide suhtes. Gale-Shapley algoritm võtab arvesse indiviidide järjestatud eelistusi ning koostab nende kriteeriumite põhjal optimaalseid gruppe. [16]

Gale-Shapley algoritmi implementeerimine töö autorite arendatud komplekteerimisfunktsiooniga rakendusse võiks leida seega kasutust erinevates Eesti koolides, kus tekib vajadus luua tõhusaid meeskondi ning hiljem analüüsida tulemusi. Selline vajadus võib esineda näiteks erinevate mälumängude või viktoriinide tiimide koostamisel.

Töö autorite projekti kõige prioriteetsem edasiarendus lähiajal on kasutajamugavuse parandamine. Suurt tähelepanu tuleb pöörata UI/UX disainile, arvestades, et see jäi praeguses arendusprotsessis rohkem tagaplaanile, kuna tähtsam eesmärk oli kogu vajalik funktsionaalsus esimese etapi alguseks kliendini toimetada. Teiseks edasiarenduseks, mis muudaks kliendi jaoks kogu protsessi veel kiiremaks ja mugavamaks, oleks liidese integreerimine, mis sisestaks masinloetavad tulemused otse loodud rakendusse. Kindlasti tuleb arvestada, et see eeldab kliendilt samuti masinloetavat tulemuste mõõtmist.

5.6 Teostatud tööde kirjeldus

Järgnevalt on esitatud töö autorite ajalogi Toggli keskkonnast (Tabel 1) ning *commit*'id GitLabi projektis on toodud Lisas 7. Kindlasti vajab siinkohal mainimist, et mitmel päeval programmeerisid tiimiliikmed koos ühest arvutist ning tegid *commit*'e ühe nime alt, mistõttu *commit*'ide arv on ebavõrdne.

Tabel 1. Meeskonnaliikmete ajaline panus.

	Analüüs (uurimistöö + dokument)	**Arendus**	**Koosolekud**	**Muu**	**Kokku**
Kadri-Liis	59h	162h	7h	10h	238h
Henri	38h	189h	7h	10h	244h

5.7 Hinnang projekti teostamise protsessi kohta

Projekti põhiosa teostati 12 nädala vältel, 29. jaanuarist kuni 19. aprillini. Projektijuhtimine toimus tiimisiseselt vastastikuselt ning mõlemad liikmed koostasid *task*'e ja kaalusid ning analüüsisid erinevaid lahenduskäike, et leida kõige optimaalsem neist. Ülesannete jaotamisel lähtuti eelkõige meeskonnaliikmete enda soovidest ja tugevustest kuid vahel võtsid autorid ette ka ebamugavamaid väljakutseid ja ülesandeid, et arendada oma nõrgemaid külgi.

Nii projekti alustamisel kui ka lõpufaasis said meeskonnaliikmed olulisi nõuandeid ja näpunäiteid juhendajalt ning projekti keskel toimus suhtlus rohkem kliendiga. Meeskonnasiseselt toimusid iganädalased koosolekud ja infovahetus oli sujuv. Omavaheliseks suhtluseks kasutati Messengeri, juhendajaga suheldi läbi MS Teamsi ja kliendiga meili teel ja Kohilas kohapeal käies. Mitmel korral harrastasid liikmed paarisprogrammeerimist. Kui esines kriitilisemaid olukordi, näiteks peale esimest etappi selgunud vajalike lisafunktsionaalsustega, leiti kiiresti lahendus ja viidi koheselt läbi muudatused.

Tiimisiseselt oli tihe ja efektiivne suhtlus ning projekti eesmärgist ja loodavast lahendusest saadi üheselt aru. Ei tekkinud segadust, et kummalgi tiimiliikmel oleks märkimisväärselt erinev visioon arendatavast rakendusest.

Protsessi kitsaskohaks oli suhtlus juhendajaga, kuna algselt ühendust võttes olid meeskonnaliikmed veendunud, et iga paari nädala tagant tehakse lühike koosolek, kus arutatakse, mis on tehtud ning kus saaks teha täiendusi. Küll aga kulges protsess suuresti omaette rakendust arendades, kuna autorid uskusid, et saavad iseseisvalt kõik situatsioonid lahendatud, mis enamjaolt nii oligi. Hiljem tundsid autorid, et oleks võinud rohkem juhendajaga kontakti luua, sest nõuanded oleksid protsessi käigus vägagi kasuks

tulnud. Projekti lõpufaasis saadi hulganisti näpunäiteid juhendajalt, kuid kindlasti oleks saanud need kõik tunduvalt varem realiseerida.

Teise kitsaskohana võib välja tuua kasutajaliidese, kuna see jäi projekti käigus pigem algeliseks, sest tuli järgida kliendi kodulehe disaini, mis vajab kliendi sõnul samuti uuendamist. Rõhk oli funktsionaalsuste loomisel, et see õigeaegselt kasutusse jõuaks.

Autorid on väga rahul valitud juhendajaga, kes oli alati kiiresti kättesaadav ning oskas iga küsimuse ja teema kohta anda otstarbekat nõu. TalTech kui teine oluline taustajõud võimaldas ligi pääseda erinevatele allikatele, mille eest muidu oleks pidanud maksma.

Autorid leiavad, et projekt kulges üldiselt sujuvalt ja probleemideta ning kliendile lubatud komponendid valmisid alati õigeaegselt. Mõlema töö autori individuaalne eneseanalüüs on toodud Lisades 8 ja 9.

5.8 Meeskondlik hinnang

Töö autorid leidsid, et kogu protsess sujus edukalt ning mõlemad tiimiliikmed panustasid töösse võrdselt (Tabel 2).

Tabel 2. Meeskondlik hinnang.

	Kadri-Liis	**Henri**
Kadri-Liis	X	0
Henri	0	X

6 Kokkuvõte

Töö eesmärk oli arendada efektiivne, automatiseeritud ja kasutajasõbralik süsteem, mis hõlbustaks Kohila pikamaajooksu sarja registreerunute haldamist, jooksude komplekteerimist ning tulemuste sisestamist ja eksportimist.

Enne projekti algust oli Kohila Spordikeskuse pikamaajooksu sarja registreerimisprotsess põhiliselt manuaalne ja ajakulukas, suurendades vigade tekkimise riski. Registreerimine käis läbi meili saatmise või helistamise ning kogu ülejäänud protsess viidi läbi Excelis käsitsi. Eesmärgi täitmiseks analüüsisid autorid erinevaid Eestis kasutatavaid lahendusi ning viisid läbi intervjuud kohalike spordiklubidega, et mõista nende vajadusi ja kogemusi. Autorid valisid sobivad tööriistad ja tehnoloogiad rakenduse arendamiseks ning viisid läbi arendusprotsessi.

Tulemusena loodi süsteem, mis automatiseerib nii registreerimise, komplekteerimise kui ka jooksude ja tulemuste eksportimise protsessi. Süsteem võimaldab osalejatel näha oma tulemusi kohe, kui tulemused on salvestatud, ning ei pea ootama, kuni tulemused Eesti Kergejõustikuliidu lehele ilmuvad.

Analüüs näitas, et uus rakendus vastab nii kliendi kui ka kasutajate ootustele, arvestades, et märkimisväärselt suurem osa esimestel etappidel osalejatest eelregistreeris end läbi loodud registreerimissüsteemi ning ainult paar üksikut osalejat saatis meili. Edasiarenduse poolest on autoritel plaanis esmalt parandada kasutajaliidese disaini ja üldist kasutajamugavust.

Kasutatud kirjandus

[1] Maraton100. „Eesti jooksuvõistlused on populaarsemad kui kunagi varem“ [www] https://www.marathon100.com/uudised/vaata/eesti-jooksuvoistlused-on-populaarsemad-kui-kunagi-varem Kasutatud: 18.04.2024.

[2] Kohila Spordikeskus. *Tulemused*. [www] https://www.kohilasport.ee/tulemused. Kasutatud: 12.04.2024.

[3] Sportity. *Sportity App*. [www] https://www.sportity.com/app Kasutatud: 02.05.2024.

[4] Eventbrite. *Home*. [www] https://www.eventbrite.com. Kasutatud: 02.05.2024.

[5] Race Entry. *Home*. [www] https://www.raceentry.com. Kasutatud: 02.05.2024.

[6] D. Rani, R. K. Ranjan, „A Comparative Study of SaaS, PaaS and IaaS in Cloud Computing,“ *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, 2014. Loetud aadressil: 03.05.2024.

[7] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, „Agile Software Development Methods: Review and Analysis,“ *VTT Publications*, vol. 478. Loetud aadressil: https://arxiv.org/pdf/1709.08439. Kasutatud: 03.05.2024.

[8] R. C. Martin, *Clean Code*. 1st ed. Upper Saddle River, NJ, US: Pearson, 2008. Kasutatud: 14.04.2024.

[9] L. De Lauretis, „From Monolithic Architecture to Microservices Architecture,“ *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW),* pp. 93-96, 2019, doi: 10.1109/ISSREW.2019.00050. Kasutatud: 03.05.2024.

[10] K. Gos and W. Zabierowski, „The Comparison of Microservice and Monolithic Architecture,“ *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pp. 150-153, 2020, doi: 10.1109/MEMSTECH49584.2020.9109514. Kasutatud: 03.05.2024.

[11] S. Deshmukh, D. Mane and A. Retawade, „Building a Single Page Application Web Front-end for E-Learning site,“ *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 985-987, 2019, doi: 10.1109/ICCMC.2019.8819703. Kasutatud: 02.05.2024.

[12] A. Leff and J. T. Rayfield, „Web-application development using the Model/View/Controller design pattern,“ *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, Seattle, WA, USA, 2001, pp. 118-127, doi: 10.1109/EDOC.2001.950428. Kasutatud: 03.05.2024.

[13] Ehsan, Adeel, Mohammed Ahmad M. E. Abuhaliqa, Cagatay Catal, and Deepti Mishra, „RESTful API Testing Methodologies: Rationale, Challenges, and Solution, Directions,“ *Applied Sciences* 12, no. 9: 4369, 2022, doi: https://doi.org/10.3390/app12094369. Kasutatud: 10.05.2024.

[14] Y. Liu jt. „Morest: Model-based RESTful API Testing with Execution Feedback” *44th International Conference on Software Engineering* Loetud aadressil: https://arxiv.org/pdf/2204.12148. Kasutatud: 12.05.2024.

[15] Keamk. *Home*. [www] https://www.keamk.com Kasutatud: 12.05.2024.

[16] D. Meulbroek, D. Ferguson, M. Ohland and F. Berry, „Forming More Effective Teams Using CATME TeamMaker and the Gale-Shapley Algorithm,“ *2019 IEEE Frontiers in Education Conference (FIE)*, Covington, KY, USA, pp. 1-5, 2019, doi: 10.1109/FIE43999.2019.9028552. Kasutatud: 12.05.2024.

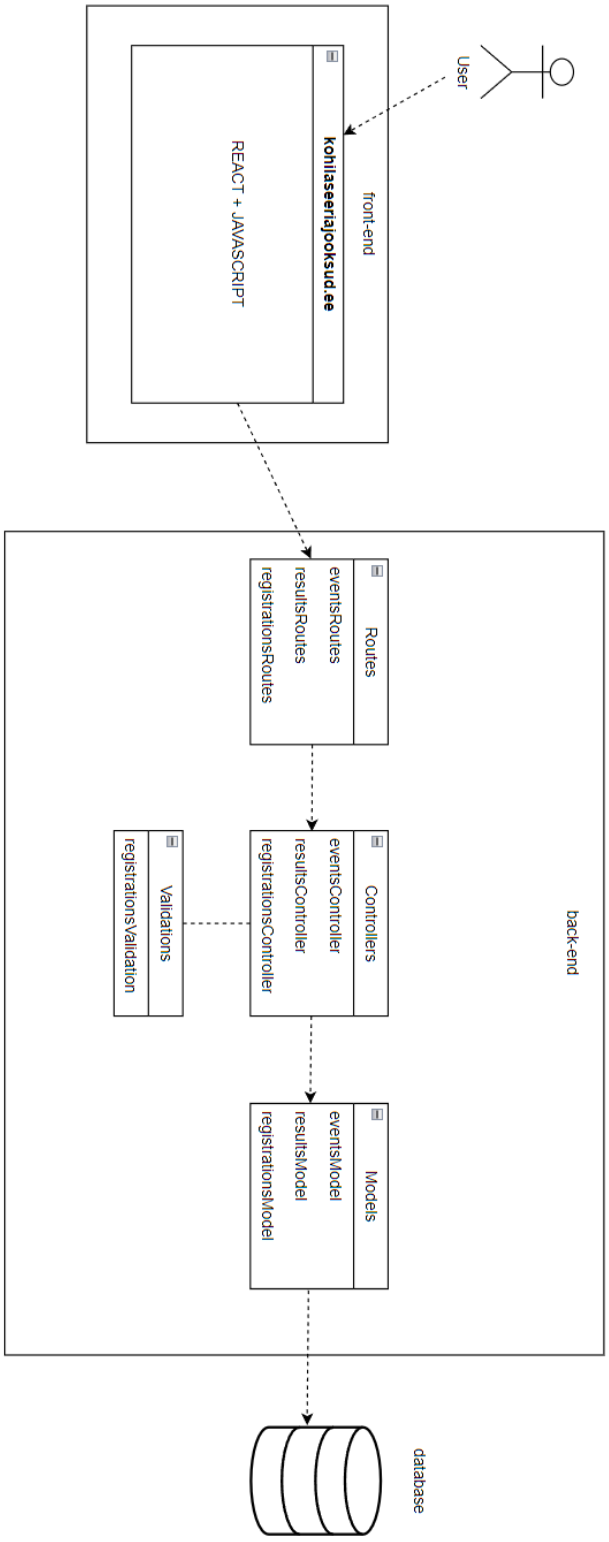
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Meie, Kadri-Liis Kolk ja Henri Haugas

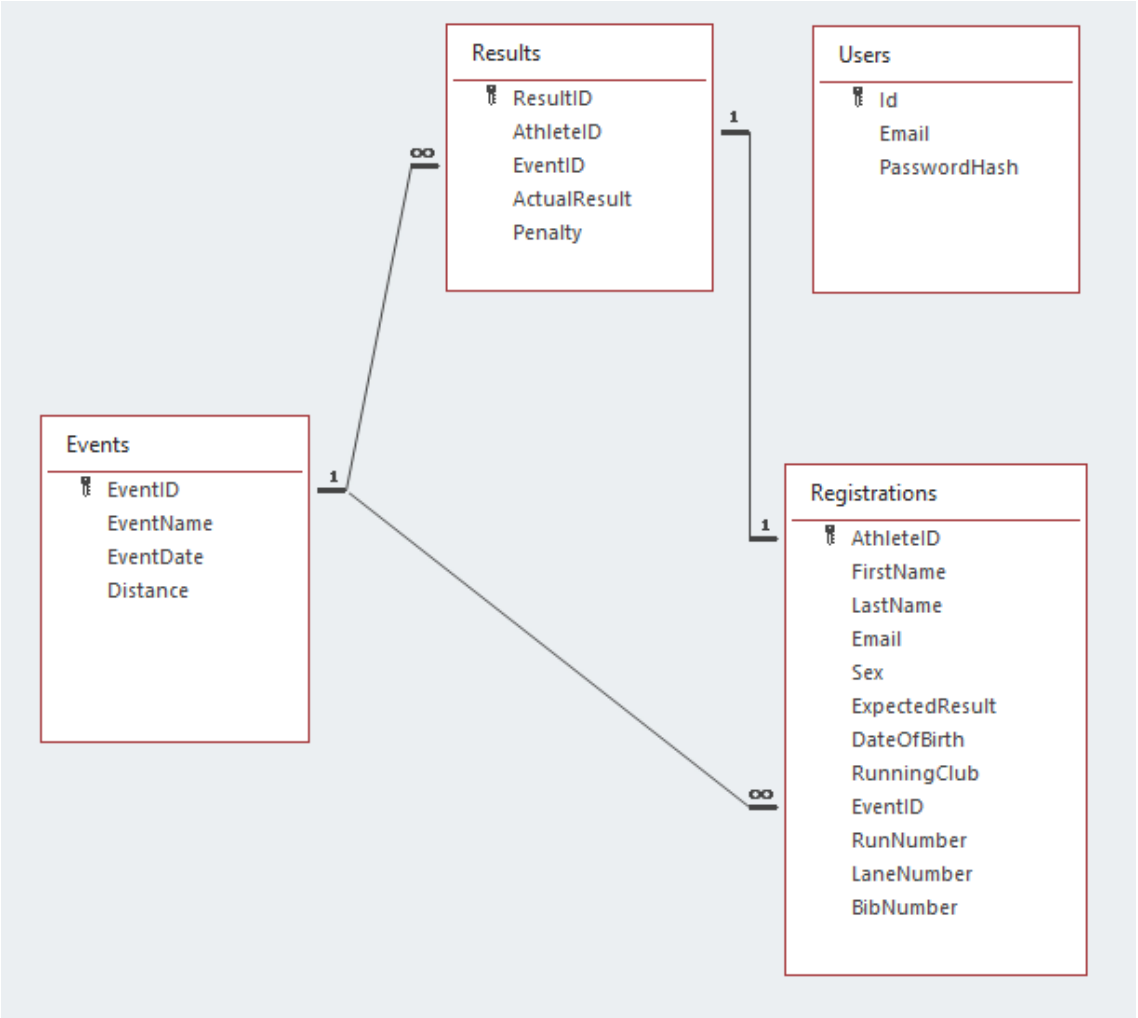
1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Kohila pikamaajooksu sarja registreerunute haldamise ja jooksude komplekteerimise süsteem“, mille juhendaja on Tarvo Treier
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna 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. Oleme teadlikud, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autoritele.
3. Kinnitame, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

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 - Rakenduse arhitektuur



Lisa 3 – Andmebaasi diagramm



Lisa 4 – Intervjuu küsimused

- Mida kasutavad võistlejad teie võistlustele registreerimiseks (mõni spetsiifiline programm, Google Forms, meili ja/või telefoni teel vms)?
- Kas kogu protsessi vältel (registreeringu kättesaamisest tulemuste printimiseni) peate käistsi võistlejate andmed kuhugi ümber kirjutama?
    - Kui jah, siis kirjeldage, mis etapis?
- Kas kasutate jooksude komplekteerimiseks mõnda programmi või teete seda käsitsi (paberil, Excel/Google Sheets vms)?
    - Kui kasutate programmi, siis millist? Kirjeldage protsessi lähemalt.
- Kas olete puudust tundnud rakendusest, mis seoks nii registreerimise, jooksude komplekteerimise, tulemuste sisestamise kui ka protokolli väljastamise ühte rakendusse?

Lisa 5 – Komplekteeritud jooksude dokument eeldatavate aegadega

JOOKS 1

1	4	RAUNO PISA	1.59,00
2	5	JESSE LAUTER	1.59,00
3	7	RASMUS KÄSPER	1.59,00
4	9	JOONAS MATTIAS LAUR	2.00,86
5	14	HUGO SÄREV	2.04,00
6	15	UKU TROLLA	2.04,00
7	17	DANEL TAUR	2.04,00

JOOKS 2

1	18	KRISTO IVASK	2.05,00
2	20	HENRI HAUGAS	2.07,50
3	25	KARL ALEKSANDER LIBA	2.10,00
4	26	LAURI TANNER	2.10,00
5	28	RAIDO RASPEL	2.12,00
6	29	MARCO KAHU	2.12,00

JOOKS 3

1	41	RAIN JANO	2.13,00
2	31	KARMEL JANO	2.14,00
3	32	MARLEEN KOLL	2.15,00
4	42	CARRY SALUSTE	2.15,00
5	44	JAAN REINOK	2.15,00
6	49	ELION SÕBER	2.18,00
7	51	KEVIN RAUDSEPP	2.19,00
8	34	BERIT GUZAS	2.19,00

JOOKS 4

1	52	ANDREAS ESNAR	2.19,00
2	35	ANNA MARIA RASPEL	2.20,00
3	53	KAIN VÄLJAOTS	2.20,00
4	54	ERKKI HUMMAL	2.20,00
5	55	RIKO AUSMEES	2.20,20
6	36	LAURA-LIISA VÄLI	2.22,00
7	57	MAREK TÕNISMÄE	2.22,00
8	59	MIHKEL MALV	2.22,00

JOOKS 5

1	60	RANDEL KUUS	2.24,00
2		KATERYNA FOMICHOVA	2.24,00
3	39	GERDA KADAK	2.25,00
4	40	MEI MARGARET TATTER	2.25,00
5	61	HELLA TÕNTS	2.27,00
6	64	BERIT KANT	2.29,00
7	65	MARTIN KÄSPER	2.30,00
8	66	KARMEN LANGI	2.30,00

JOOKS 6

1	13	MERLIN TOIRK	2.34,00

2	67	MIRJAM PIIK	2.35,00
3	68	KÄTLIN KOPLI	2.39,00
4	69	AHTI SEIER	2.40,00
5	30	ANDRUS RÜNDAL	2.40,00
6	76	VILLU ZIRNASK	2.48,00
7	77	DIRK-EDRIK TORI	2.50,00
8	83	VANESSA SIMUSTE	2.50,00

JOOKS 7

1	84	VALERI KÄÄPA	2.50,00
2	88	JAAN LOMP	2.55,00
3	90	IVAR RAIG	2.59,00
4	91	AKSEL WEILER	2.59,00
5	92	GUSTAV KIIVER	3.00,00
6	93	VIIVI KÄÄPA	3.00,00
7	10	HEIKI SASSIAN	3.10,00
8	94	HUGO BERT VAHEMETS	3.20,00

JOOKS 8

1	14	OLEV MITT	00,00
2	16	SIRLE SUVISTE	00,00
3	95	OSKAR MALV	3.30,00
4	98	JUHAN PAABSTEL	3.59,00
5	15	MARGUS SIKK	00,00
6	100	ARNE KARMING	5.00,00

Lisa 6 – Tulemuste protokoll

Kohila
Pikamaajooksu sari
2 Etapp
800m

15.05.2024
Kohila staadion

N -19	Koht	Nimi	Sünniaeg	Klubi	Aeg	Punktid
	1	Karmel JANO	20.01.2007	Rapla jooksuklubi	2.21,59	12
	2	Berit GUZAS	09.12.2005	Nõmme KJK	2.21,78	10
	3	Berit KANT	23.02.2008	Rapla Jooksuklubi	2.30,78	8
	4	Gerda KADAK	25.01.2009	Tallinna Spordiakadeemia	2.32,55	7
	5	Mei Margaret TATTER	05.02.2007	SK Fortis	2.33,76	6
	6	Merlin TOIRK	10.09.2011	Rapla Jooksuklubi	2.38,59	5
	7	Kätlin KOPLI	09.09.2006	Rapla Jooksuklubi	2.55,69	4
N 20-29	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Marleen KOLL	21.03.2005	Sparta SS	2.17,38	12
	2	Anna Maria RASPEL	01.03.2005	Rae Kergejõustik	2.22,90	10
	3	Laura-Liisa VÄLI	23.09.2003	Rapla Jooksuklubi	2.23,56	8
	4	Hella TÕNTS	21.09.2004	Tallinna Spordiakadeemia	2.26,17	7
	5	Karmen LANGI	13.03.2002		2.34,18	6
	6	Vanessa SIMUSTE	29.09.2004	Tallinna Spordiakadeemia	2.56,17	5
N 30-39	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Kateryna FOMICHOVA	15.03.1993	Täppsportlased	2.16,18	12
N 40-49	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Mirjam PIIK	21.12.1977	Täppsportlased	2.40,44	12
	2	Sirle SUVISTE	29.12.1981		3.29,90	10
N 60+	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Viivi KÄÄPA	20.06.1962	Läänemaa JK VII&VA	3.15,89	12
M -19	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Jesse LAUTER	01.03.2006	Rapla Jooksuklubi	2.01,93	12
	2	Uku TROLLA	19.06.2007		2.14,05	10
	3	Andreas ESNAR	09.01.2009	UP Sport	2.15,10	8
	4	Hugo SÄREV	30.01.2007	Rapla Jooksuklubi	2.16,52	7
	5	Riko AUSMEES	06.01.2009	UP Sport	2.21,02	6
	6	Karl Aleksander LIBA	14.11.2007	Kurtna kool	2.21,95	5
	7	Mihkel MALV	26.01.2008		2.24,53	4
	8	Kevin RAUDSEPP	10.11.2009	Rapla Jooksuklubi	2.36,11	3
	9	Randel KUUS	29.04.2010	Rapla Jooksuklubi	2.40,01	2
	10	Dirk-Edrik TORI	05.10.2012		2.47,75	1
	11	Hugo Bert VAHEMETS	24.08.2012		2.50,82	0,5
	12	Gustav KIIVER	09.08.2015	Kohila Püsivus	2.53,55	0,5
	13	Oskar MALV	23.02.2011	Kohila SK	3.01,73	0,5
M 20-29	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Joonas Mattias LAUR	01.06.1999	Nõmme KJK	1.59,34	12
	2	Rasmus KÄSPER	08.09.2002	Tallinna Spordiakadeemia	1.59,35	10
	3	Rauno PISA	02.01.2002	Rapla Jooksuklubi	2.05,42	8
	4	Henri HAUGAS	05.10.2002	Tallinna Spordiakadeemia	2.12,99	7

M 30-39	Koht	Nimi	Sünniaeg	Klubi	Aeg	Punktid
	1	Kristo IVASK	16.12.1986		2.10,69	12
	2	Marco KAHU	01.02.1989	Sparta	2.14,92	10
	3	jaan REINOK	16.08.1990		2.17,85	8
	4	Carry SALUSTE	06.11.1990	Copper Trainings	2.19,98	7
	5	Elion SÕBER	04.07.1991		2.27,09	6
	6	Margus SIKK	11.08.1988		2.48,23	5
M 40-49	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Danel TAUR	28.03.1976	Täppsportlased	2.04,66	12
	2	Lauri TANNER	23.12.1978	Kadarbiku Köögivili	2.13,38	10
	3	Marek TÕNISMÄE	08.10.1975	Täppsportlased	2.20,55	8
	4	Kain VÄLJAOTS	29.06.1974		2.20,61	7
	5	Ahti SEIER	01.02.1982		2.32,14	6
M 50-59	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Raido RASPEL	07.10.1967	Treeningprtner Sportland	2.14,22	12
	2	Rain JANO	14.06.1973	Jk Sarma	2.18,42	10
	3	Erkki HUMMAL	02.09.1973	SPARTA	2.21,64	8
	4	Villu ZIRNASK	05.08.1966		2.45,10	7
	5	Aksel WEILER	17.09.1972	Rapla jooksuklubi	2.58,22	6
M 60+	**Koht**	**Nimi**	**Sünniaeg**	**Klubi**	**Aeg**	**Punktid**
	1	Martin KÄSPER	11.04.1964		2.27,13	12
	2	Andrus RÜNDAL	21.01.1964	Rapla SVK	2.44,51	10
	3	Valeri KÄÄPA	02.08.1958	Läänemaa JK VII&VA	2.52,25	8
	4	IVAR RAIG	25.03.1953	TSVK	2.53,95	7
	5	Jaan LOMP	11.11.1944		2.57,42	6
	6	Olev MITT	02.07.1961		3.02,00	5
	7	Heiki SASSIAN	09.09.1961	Rapla SVK	3.06,52	4
	8	Juhan PAABSTEL	05.05.1959		3.53,67	3
	9	Arne KARMING	20.11.1951	Rapla SVK	4.16,23	2

Peakohtunik
Aavo Kergand

Lisa 7 – Teostatud tööde kirjeldus

- Henri Haugas 2024-05-11 04:06:45 new database connectionLimit
- Henri Haugas 2024-05-11 03:27:18 fix pool is closed error
- Henri Haugas 2024-05-11 03:24:07 this should fix
- Henri Haugas 2024-05-11 03:21:05 api blew up now fixing
- Henri Haugas 2024-05-11 03:14:29 API pool.end and acquireTimeout test
- Henri Haugas 2024-05-11 02:58:46 nevermind dont need console log
- Henri Haugas 2024-05-11 02:58:01 testing
- Henri Haugas 2024-05-11 02:54:29 tulemused visible only after 2 days of comp
- Henri Haugas 2024-05-11 02:41:47 results veiwable for everyone
- Henri Haugas 2024-05-10 23:33:19 Merge branch 'testAfterFirstEvent' into 'main'
- Henri Haugas 2024-05-11 02:32:17 changes after first competition
- Henri Haugas 2024-05-09 10:41:00 men age group fix
- Henri Haugas 2024-05-09 10:26:56 missing age group
- Henri Haugas 2024-05-08 12:36:44 total number of registrations
- Henri Haugas 2024-05-08 09:02:09 testing without braces
- Henri Haugas 2024-05-08 08:51:43 display total number of registrations
- Henri Haugas 2024-05-06 23:43:18 for pictures
- Henri Haugas 2024-05-05 19:27:55 updated logging
- Henri Haugas 2024-05-04 14:32:48 logging registrations
- Henri Haugas 2024-05-02 11:06:19 if first connection to server fails, try again
- Henri Haugas 2024-05-02 00:02:40 retry logic
- Henri Haugas 2024-05-01 21:33:54 database pool configure settings
- Henri Haugas 2024-05-01 19:52:09 show me error
- Henri Haugas 2024-04-19 00:57:34 final changes
- Henri Haugas 2024-04-19 00:45:10 lower case
- Henri Haugas 2024-04-19 00:43:47 wrong +1
- Henri Haugas 2024-04-19 00:41:08 etapid with roman numbers
- Henri Haugas 2024-04-19 00:24:51 input change
- Henri Haugas 2024-04-19 00:14:51 input arrows change
- Henri Haugas 2024-04-19 00:08:56 reg css

- Henri Haugas 2024-04-19 00:03:03 reg css
- Henri Haugas 2024-04-19 00:01:12 reg css
- Henri Haugas 2024-04-18 23:57:44 reg css
- Henri Haugas 2024-04-18 23:56:29 reg css
- Henri Haugas 2024-04-18 23:46:30 registration page css
- Henri Haugas 2024-04-18 23:26:34 JWT from env
- Henri Haugas 2024-04-18 23:20:48 await async pool test
- Henri Haugas 2024-04-18 23:10:56 testing pool connection
- Henri Haugas 2024-04-18 18:46:42 testing sql connection
- Kadri-Liis 2024-04-16 02:31:19 reg
- Kadri-Liis 2024-04-16 02:28:39 reg
- Kadri-Liis 2024-04-16 02:27:19 reg
- Kadri-Liis 2024-04-16 02:22:40 reg css
- Kadri-Liis 2024-04-16 02:19:12 reg css
- Kadri-Liis 2024-04-16 02:14:08 reg css
- Kadri-Liis 2024-04-16 02:03:05 registration css
- Kadri-Liis 2024-04-16 01:57:19 registration css
- Kadri-Liis 2024-04-15 17:12:15 token
- Kadri-Liis 2024-04-15 17:09:25 logout cookie
- Kadri-Liis 2024-04-15 17:00:13 logout
- Kadri-Liis 2024-04-15 16:57:48 logout
- Kadri-Liis 2024-04-15 16:54:26 logout
- Kadri-Liis 2024-04-15 16:39:45 login, logout
- Kadri-Liis 2024-04-15 16:23:30 login
- Kadri-Liis 2024-04-15 16:11:36 logout change
- Kadri-Liis 2024-04-15 15:58:11 login
- Kadri-Liis 2024-04-15 15:50:14 revert login changes
- Kadri-Liis 2024-04-15 15:49:34 login revert changes
- Kadri-Liis 2024-04-15 15:37:20 login changes
- Kadri-Liis 2024-04-15 15:24:45 login changes
- Henri Haugas 2024-04-15 14:27:42 new JWT parameters test
- Henri Haugas 2024-04-15 14:09:10 testing JWT
- Henri Haugas 2024-04-15 14:01:19 testing new hash
- Henri Haugas 2024-04-15 13:51:22 testing login 3

- Henri Haugas 2024-04-15 13:47:55 testing login2
- Henri Haugas 2024-04-15 13:43:56 testing login
- Kadri-Liis 2024-04-15 01:46:06 login try 3
- Kadri-Liis 2024-04-15 01:18:16 login vol 2
- Kadri-Liis 2024-04-15 00:48:48 log in change vol 1
- Kadri-Liis 2024-04-15 00:39:07 css fixes
- Kadri-Liis 2024-04-15 00:14:04 registration css
- Kadri-Liis 2024-04-15 00:10:45 registration css fix
- Henri Haugas 2024-04-14 23:56:36 log in url fix
- Kadri-Liis 2024-04-14 23:49:11 api url
- Kadri-Liis Kolk 2024-04-14 20:35:46 Merge branch 'test' into 'main'
- Kadri-Liis 2024-04-14 23:18:17 css fixes
- Henri Haugas 2024-04-14 23:11:45 fix capital letter typo
- Henri Haugas 2024-04-14 23:06:59 update pool
- Henri Haugas 2024-04-14 22:52:08 hello okay
- Henri Haugas 2024-04-14 22:46:07 maybe this time
- Henri Haugas 2024-04-14 22:42:39 this should work
- Henri Haugas 2024-04-14 22:40:06 am i getting those env variables correctly? test 2
- Henri Haugas 2024-04-14 22:37:45 am i getting those env variables correctly?
- Henri Haugas 2024-04-14 22:30:50 added port env
- Henri Haugas 2024-04-14 22:20:59 change env names
- Henri Haugas 2024-04-14 21:41:15 new frontend link
- Henri Haugas 2024-04-14 21:09:39 new API url
- Henri Haugas 2024-04-14 17:13:12 Merge branch 'test' into 'main'
- Henri Haugas 2024-04-14 20:12:24 backend start script
- Henri Haugas 2024-04-14 17:05:46 Merge branch 'test' into 'main'
- Henri Haugas 2024-04-14 20:04:57 clean database before deploy
- Henri Haugas 2024-04-14 20:01:38 ready for deploy
- Kadri-Liis 2024-04-14 15:24:27 added user
- Kadri-Liis Kolk 2024-04-14 11:42:05 Merge branch '23-log-in' into 'test'
- Kadri-Liis 2024-04-14 14:41:27 log in for safari fixed
- Kadri-Liis 2024-04-12 23:08:29 login working
- Kadri-Liis 2024-04-11 15:01:23 login muudatused

- Kadri-Liis 2024-04-11 00:50:15 logging in almost working
- Kadri-Liis 2024-04-11 00:37:22 log in almost working
- Kadri-Liis 2024-04-09 00:08:55 started login
- Kadri-Liis Kolk 2024-04-08 21:07:51 Merge branch 'main' into '23-log-in'
- Kadri-Liis Kolk 2024-04-08 21:07:03 Merge branch 'test' into 'main'
- Kadri-Liis 2024-04-08 22:00:09 new scrollbar fix
- Kadri-Liis 2024-04-08 21:52:38 scrolling
- Kadri-Liis 2024-04-08 21:42:57 error message and scrolling
- Kadri-Liis 2024-04-08 20:20:20 error message width
- Kadri-Liis 2024-04-08 20:11:27 error message, long name bug fix
- Henri Haugas 2024-04-08 17:33:53 minor fixes and better error messages
- Henri Haugas 2024-04-08 17:20:22 minor fixes and better error messages
- Henri Haugas 2024-03-31 18:56:57 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-31 21:55:53 more bug fixes
- Henri Haugas 2024-03-30 20:10:12 bug fixes
- Henri Haugas 2024-03-24 23:44:41 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-25 01:44:04 printing runs, formatting times and bug fixes
- Henri Haugas 2024-03-24 22:19:32 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-24 22:19:02 Merge branch '34-add-run' into 'test'
- Kadri-Liis 2024-03-25 00:15:16 another changes in bib
- Kadri-Liis 2024-03-25 00:09:16 entering bib number
- Kadri-Liis 2024-03-24 22:36:48 design fixes
- Henri Haugas 2024-03-24 19:37:52 bug fixes
- Kadri-Liis 2024-03-24 15:58:26 added athlete doesn't show immediately
- Kadri-Liis 2024-03-23 14:04:11 switching order between and in containers works, but not dnd for empty container
- Kadri-Liis 2024-03-23 01:06:10 lane number, bib, dnd
- Henri Haugas 2024-03-22 21:39:05 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-22 23:37:41 online registration closes after 16, milliseconds formatting
- Henri Haugas 2024-03-21 19:41:15 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-21 19:39:57 Merge branch '47-link-to-other-competitors-for-this-event' into 'test'
- Henri Haugas 2024-03-21 21:38:58 fixed navigating to event after registering

- Henri Haugas 2024-03-20 23:01:45 fixed adding a runner
- Kadri-Liis 2024-03-20 20:47:59 adding athlete after grouping almost working
- Henri Haugas 2024-03-20 17:06:42 adding runner at grouping
- Henri Haugas 2024-03-20 16:53:26 started adding new button
- Henri Haugas 2024-03-20 12:19:47 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-20 14:19:21 times formatter, events sorter
- Kadri-Liis Kolk 2024-03-20 10:44:09 Merge branch 'test' into 'main'
- Kadri-Liis Kolk 2024-03-20 10:43:35 Merge branch '35-registered-page' into 'test'
- Kadri-Liis 2024-03-20 12:42:56 separate registered page logic changes
- Henri Haugas 2024-03-19 22:11:10 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-19 22:09:46 Merge branch '10-export-results' into 'test'
- Henri Haugas 2024-03-19 22:09:38 Merge branch 'test' into '10-export-results'
- Henri Haugas 2024-03-20 00:06:21 finished results export
- Henri Haugas 2024-03-19 16:28:06 bold
- Henri Haugas 2024-03-19 16:13:06 starting to get the correct format
- Henri Haugas 2024-03-19 15:23:52 some exporting works now
- Kadri-Liis 2024-03-16 21:07:13 separate registered page
- Kadri-Liis Kolk 2024-03-15 21:53:08 Merge branch 'test' into 'main'
- Kadri-Liis Kolk 2024-03-15 21:52:23 Merge branch '39-design-fixes' into 'test'
- Kadri-Liis 2024-03-15 23:44:22 checkbox and penalty fix
- Henri Haugas 2024-03-14 16:26:15 first testing
- Kadri-Liis 2024-03-13 22:44:47 changes in entering results, design fixes also
- Henri Haugas 2024-03-13 21:22:25 started with results exporting
- Kadri-Liis 2024-03-13 14:16:00 button aligning on registered page
- Kadri-Liis 2024-03-13 14:02:03 registration form aligning, small fixes
- Henri Haugas 2024-03-12 19:37:27 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-12 19:36:50 Merge branch '38-run-ended-status' into 'test'
- Henri Haugas 2024-03-12 21:35:30 fixed event status
- Henri Haugas 2024-03-12 19:00:00 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-12 18:59:01 Merge branch '36-backend-registration-validation-to-check-f-and-m' into 'test'
- Henri Haugas 2024-03-12 20:57:42 added gender check
- Kadri-Liis Kolk 2024-03-11 11:10:50 Merge branch 'test' into 'main'

- Kadri-Liis Kolk 2024-03-11 11:10:21 Merge branch '32-complect-runs' into 'test'
- Kadri-Liis 2024-03-11 13:08:00 changes in sorting
- Kadri-Liis 2024-03-10 23:56:48 results page, checkbox, error message
- Henri Haugas 2024-03-10 17:12:43 results age group table changes
- Kadri-Liis 2024-03-10 15:23:20 toggle button
- Kadri-Liis 2024-03-10 14:07:48 removed comments
- Kadri-Liis 2024-03-10 14:01:15 grouping working
- Kadri-Liis Kolk 2024-03-09 17:46:35 Merge branch '32-complect-runs' into 'test'
- Kadri-Liis 2024-03-09 19:38:39 fixed complecting and modal
- Kadri-Liis 2024-03-09 15:23:03 dnd save
- Kadri-Liis 2024-03-09 14:29:08 changes in complecting
- Kadri-Liis Kolk 2024-03-08 11:27:33 Merge branch 'test' into 'main'
- Kadri-Liis Kolk 2024-03-08 11:26:01 Merge branch '30-implement-designs' into 'test'
- Kadri-Liis 2024-03-08 13:22:31 fixes on registered page
- Henri Haugas 2024-03-08 00:53:02 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-08 00:51:23 Merge branch '33-results-table-based-on-age-group' into 'test'
- Henri Haugas 2024-03-08 02:50:00 results grouped by age and gender
- Henri Haugas 2024-03-07 23:16:45 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-07 23:16:17 Merge branch '31-date-problem' into 'test'
- Henri Haugas 2024-03-08 01:15:24 fixed DoB and event date problem
- Henri Haugas 2024-03-07 21:57:18 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-07 21:56:51 Merge branch '27-manual-athlete-adding' into 'test'
- Henri Haugas 2024-03-07 23:55:45 finished manual adding in registered view
- Henri Haugas 2024-03-07 16:55:21 add athlete manually in registered page
- Kadri-Liis 2024-03-06 21:04:59 fixes on registration
- Henri Haugas 2024-03-06 20:17:23 started manual athlete adding
- Henri Haugas 2024-03-05 18:02:16 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-05 18:01:34 Merge branch '29-switch-to-react-vite' into 'test'
- Henri Haugas 2024-03-05 19:58:57 switched to vite to keep away from vulnurabilities

- Henri Haugas 2024-03-05 16:34:55 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-05 16:32:53 Merge branch '11-designs' into 'test'
- Kadri-Liis 2024-03-05 18:29:51 registered, results and competition pages
- Henri Haugas 2024-03-01 15:48:15 Merge branch 'test' into 'main'
- Henri Haugas 2024-03-01 15:45:20 Merge branch '25-create-backend-validation' into 'test'
- Henri Haugas 2024-03-01 17:43:46 registration validation and correct kohila events into database
- Kadri-Liis 2024-02-27 23:14:58 competitions and registered initial design
- Kadri-Liis 2024-02-27 20:09:34 registration page changes
- Kadri-Liis 2024-02-27 18:00:06 registration form design fixes
- Kadri-Liis 2024-02-25 23:25:31 registration form design
- Henri Haugas 2024-02-26 21:41:26 Merge branch 'test' into 'main'
- Henri Haugas 2024-02-26 21:40:42 Merge branch '26-insert-results-and-results-page' into 'test'
- Henri Haugas 2024-02-26 23:39:34 insert results and results table
- Henri Haugas 2024-02-26 16:24:31 started implementing insert results and results tables
- Henri Haugas 2024-02-25 20:20:31 Merge branch 'test' into 'main'
- Henri Haugas 2024-02-25 20:19:47 Merge branch '24-registered-athletes-table-page' into 'test'
- Henri Haugas 2024-02-25 22:18:42 etappide, registreerunute ja komplekteerimise lehe algus
- Henri Haugas 2024-02-23 22:50:01 started adding registered table
- Henri Haugas 2024-02-23 13:40:51 started implementing registered tables
- Henri Haugas 2024-02-23 01:27:42 implemented new backend endpoint
- Henri Haugas 2024-02-22 22:41:46 Merge branch 'test' into 'main'
- Henri Haugas 2024-02-23 00:40:40 auto increment results
- Henri Haugas 2024-02-22 22:38:50 Merge branch '18-implement-increment-by-1' into 'main'
- Henri Haugas 2024-02-23 00:38:06 auto-increment Results
- Henri Haugas 2024-02-22 22:34:04 Merge branch 'test' into 'main'
- Henri Haugas 2024-02-22 22:33:17 Merge branch '17-create-form-for-registering-to-events' into 'test'

- Henri Haugas 2024-02-22 16:02:03 changed favicon and submit button
- Henri Haugas 2024-02-22 15:31:03 refactoring and no duplicate registrations allowed
- Henri Haugas 2024-02-18 15:51:56 validation and some features
- Henri Haugas 2024-02-18 03:10:29 working with form
- Henri Haugas 2024-02-17 02:30:00 added basic form
- Henri Haugas 2024-02-16 22:23:32 Merge branch 'test' into 'main'
- Henri Haugas 2024-02-16 22:22:13 Merge branch '18-implement-increment-by-1' into 'test'
- Henri Haugas 2024-02-17 00:19:32 added autoincrement to events and registrations
- Henri Haugas 2024-02-16 21:54:13 Merge branch 'test' into 'main'
- Henri Haugas 2024-02-16 23:53:16 fixed merge errror
- Henri Haugas 2024-02-16 21:19:09 Merge branch 'revert-0498f792' into 'test'
- Henri Haugas 2024-02-16 21:15:16 Revert "Merge branch '7-api-endpoints' into 'main'"
- Henri Haugas 2024-02-16 21:15:01 Merge branch '7-api-endpoints' into 'main'
- Henri Haugas 2024-02-15 21:58:30 finished registrations and routes
- Henri Haugas 2024-02-15 17:12:21 events CRUD ready
- Henri Haugas 2024-02-15 11:36:40 Merge branch 'test' into 'main'
- Henri Haugas 2024-02-13 16:06:57 added not null to database,email,backend get all registrations endpoint
- Henri Haugas 2024-02-13 13:33:46 namechange
- Henri Haugas 2024-02-13 13:26:51 Merge branch '5-file-structures' into 'test'
- Kadri-Liis 2024-02-12 16:34:59 fixes in folder structure
- Kadri-Liis Kolk 2024-02-12 12:37:39 Merge branch '5-file-structures' into 'test'
- Kadri-Liis 2024-02-10 21:02:57 API structure
- Kadri-Liis 2024-02-05 20:01:33 folder structures
- Kadri-Liis 2024-02-04 23:33:35 second
- Henri Haugas 2024-02-04 23:26:30 lets see if this works
- Kadri-Liis 2024-02-04 23:19:28 initial commit
- Henri Haugas 2024-02-04 22:16:33 cookies fix
- Henri Haugas 2024-02-04 21:51:27 testing
- Henri Haugas 2024-02-04 20:54:49 new database structure

- Henri Haugas 2024-02-04 19:41:12 added frontend
- Henri Haugas 2024-02-04 18:12:57 Merge branch 'main' of https://gitlab.com/henri.haugas.hh/kohila_seeriajooksud
- Henri Haugas 2024-02-04 18:12:18 gitignore test
- Henri Haugas 2024-02-04 16:09:52 Delete .env
- Henri Haugas 2024-02-04 18:09:20 ignore env
- Henri Haugas 2024-02-04 18:07:49 backend
- Henri Haugas 2024-02-04 16:02:31 Initial commit

Lisa 8 – Kadri-Liisi eneseanalüüs

Projekti raames tegelesin veidi suuremal määral *front-end* arendusega, kui *back-end* arendusega, kuid sellegipoolest puutusin kokku mõlemaga ja sain arendada end mitmest küljest. Väiksemaid funktsionaalsustega seotud muudatusi tegin igas vaates, kuid minu peamised ülesanded olid järgnevad:

- kasutajaliidese disaini loomine rakendusele ning selle lisamine rakendusele CSSi ja HTMLi abil;
- sisse logimise ja autentimise funktsionaalsuse loomine kliendile, kasutades *JWT*'d;
- komplekteerimise erinevad funktsioonid, mis hõlmasid manuaalset võistlejate ümber tõstmist ehk *drag-and-drop*'i (jooksjat on võimalik liigutada samas jooksus teise kohta ning samuti ühest jooksust teise), rinnanumbrite sisestamise võimalust, rajanumbrite loogikat, jooksja lisamist ning jooksu lisamist (koostöös tiimikaaslasega);
- registreerunute vaate loomine ja pärimine API-st;
- kasutusjuhendi loomine kliendile;
- intervjuude läbi viimine ja analüüsimine.

Suurimaks väljakutseks kujunes ootamatult *drag-and-drop*, kuna sellega ilmnes mitmeid erandlikke olukordi, mis tuli läbi mõelda, et klient saaks kõikvõimalikel viisidel võistlejate asukohti muuta. Omajagu väljakutset esitas ka React raamistiku ja JavaScripti selgeks tegemine iseendale, kuna pole neid varasemalt õppeainetes kasutanud ning seetõttu võin julgelt väita, et see oli samuti kõige arendavam osa.

Lihtsam osa oli minu jaoks disaini lisamine kasutajaliidesele, kuna erinevalt Reactist ja JavaScriptist olin CSSi ja HTMLiga varasemalt kokku puutunud. Tunnen, et oleksin võinud rohkem aega panustada disaini loomisele, kuid nagu juba eelnevalt mainitud, oli meie jaoks prioriteetsem saada kogu funktsionaalsus toimima ning kasutajaliidese disainiga tegeleme edasiarenduse käigus. Siinkohal mainiksin, et oleksin soovinud suuremat ettevalmistust UI/UX disaini puhul kooli poolt ka, sest seda teemat puudutati üsna õrnalt, kuid kindlasti oleksid sellealased teadmised ära kulunud.

Koolist saadud teadmistest tulid kindlasti suuresti kasuks API-de kasutus ning sama võib öelda Vue raamistiku kasutuse kohta, sest hoolimata sellest, et Vue asemel kasutasime Reacti, siis need toimivad üpris sarnaselt. Lisaks mängisid suurt rolli agiilse arenduse ja sealhulgas Scrumiga seotud teadmised, et osata arendusprotsessi tõhusalt planeerida ning läbi viia. Oluline osa oli andmebaaside tundmisel, et luua andmebaas ja koostada päringuid. Viimaks tooksin välja testimise läbi Postmani. Kuigi minu tiimikaaslane tegeles Postmanis testimisega, siis arendusprotsessi käigus kasutasime testimiseks ka VS Code'i laiendust nimega Thunder Client, mis toimib identselt Postmaniga.

Lisa 9 – Henri eneseanalüüs

Rakenduse loomisel tegelesin põhiliselt rakenduse andmebaasi ja *back-end*'i loomise, juurutamise ja testimisega. *Front-end* disaini poolt tegin pigem vähem aga funktsionaalsusi kirjutasin. Suuremate *front-end* funktsionaalsuste hulka kuulusid:

- kasutaja sisendite valideerimised;
- ajaliste ja kuupäevaliste andmete vorming;
- jooksude ja tulemuste eksportimine prinditavaks Word dokumendiks;
- jooksude automaatne komplekteerimine aegade alusel;
- vaadete vaheline navigeerimine;

Kõige raskemaks proovikiviks osutus alguses React raamistikuga kohanemine, olles varasemalt tööl JavaScripti kirjutanud ja koolis Vue raamistikuga kokku puutunud oli Reacti olekuhaldus (*state management*) siiski nendest erinev ning tekitas omajagu segadust. Samas oli see arendav ja aitas paremini selgeks saada DOM (*Document Object Model*) kontseptsiooni. Lisaks oli üks raskemaid aspekte kogu rakenduse kasutuselevõtt, sest algul oli probleeme stabiilse andmebaasi ja *back-end*'i vahelise ühenduse loomisega. Peale mitmeid tunde dokumentatsiooni ja foorumite lugemist sai ka see mure lahenduse.

Lihtsamateks osadeks olid API lõpp-punktide kirjutamine, nende testimine Postmaniga ja andmebaasi päringute tegemine, sest ülikoolis õpitu on mind nendeks ülesanneteks hästi ette valmistanud.

Arendamise käigus oleksin võinud rohkem rõhku panna täpsemate veateadete loomisele, sest tihti tuli protsessi jooksul ette olukordi, kus parema kirjeldusega error kiirendanud vigade parandamist.