TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Emili Kriisa 222550IAIB

Avaliku rakendusliidese loomine Eesti Jalgpalli Liidule

Bakalaureusetöö

Juhendaja: Siim Rebane
BSc

Kaasjuhendaja: Argo Linnaste
BSc

Tallinn 2025

Autorideklaratsioon

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

Autor: Emili Kriisa

04.06.2025

Annotatsioon

Eesti Jalgpalli Liit on organisatsioon, mis haldab kohalikke jalgpallikoondisi ja Eesti esindusmeeskonda ning korraldab kohalikke jalgpallimeistrivõistlusi. Jalgpalliklubidele, uudisteportaalidele ja muudele sidusrühmadele on igapäevaseks toimimiseks vajalik pidev ligipääs ajakohasele informatsioonile koondiste, mängude, mängijate jm kohta. Eesti Jalgpalli Liit on seega vastutav sellise infovahetuse võimaluse pakkumise eest osapooltele.

Käesoleva bakalaureusetöö eesmärk on luua Eesti Jalgpalli Liidule rakendustarkvara liides ehk API, mida on võimalik kasutada kõikidel koostööpartneritel päringute tegemiseks tarviliku informatsiooni kohta. Tulemuseks on ühtne teenus, mis on valmis avalikuks kasutamiseks ja võimaldab lihtsat ning sujuvat andmevahetust. Üldiseks eesmärgiks on suurendada läbipaistvust ja lihtsustada andmete kättesaadavust, et toetada jalgpalli päevakohasust Eestis.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 6 peatükki, 7 joonist.

Abstract

Creation of Public Application Programming Interface for Estonian Football Association

Estonian Football Association is an organization that manages local football teams and the Estonian national football team. The association is also responsible for organizing all local football championships, ranging from youth leagues to professional tournaments, and organizing national matches taking place in Estonia. In addition, it plays a key role in promoting the sport across different levels and ensuring the long-term development of football in the country.

For football clubs, news portals and other stakeholders, continuous access to up-to-date information about teams, players, matches etc. is necessary for completing their daily operations. The Estonian Football Association is therefore responsible for providing a possibility for such information exchange to the parties involved and ensuring smooth interactions with stakeholders. ö The aim of this thesis is to create an application programming interface (API) for the Estonian Football Association which can be used by all stakeholders who wish to make queries to obtain information that is relevant to them. The resulting product is a unified service that is ready for public use, which enables simple and smooth data exchange. This solution also lays the groundwork for the organization's future digital services that may build upon the same infrastructure. The general goal is to increase transparency and simplify data accessibility to support the overall relevance of football in Estonia.

The thesis is in Estonian and contains 33 pages of text, 6 chapters, 7 figures.

Lühendite ja mõistete sõnastik

AES-256-CBC	Võtmega krüpteerimisalgoritm (*Advanced Encryption Standard with a 256-bit key in Cipher Block Chaining (CBC) mode*)
API	Rakendusliides, programmiliides, rakendustarkvara liides (*Application Programming Interface*)
CRUD	Püsiliku andmebaasi põhioperatsioonid (*Create, read, update, delete*)
FIFA	Rahvusvaheline Jalgpalliliit, Rahvusvaheline Jalgpalliliitude Föderatsioon (pr. k. *Fédération internationale de football association*)
HTTPS	Turvaline hüperteksti edastusprotokoll (*Hypertext Transfer Protocol Secure*)
JSON	JavaScripti programmeerimiskeelel põhinev andmevahetusvorming (*JavaScript Object Notation*)
PHP	Programmeerimiskeel (*Hypertext Preprocessor*)
REST	Tarkvaraarhitektuur (*Representational state transfer*)
SQL	Struktuurpäringukeel (*Structured Query Language*)
UEFA	Euroopa Jalgpalliliit (*Union of European Football Associations*)
XML	Laiendatav märgistuskeel (*Extensible Markup Language*)

Sisukord

Jooniste loetelu

1 Sissejuhatus

Iga päevaga aina rohkem digitaliseeruvas maailmas on andmevahetuse tähtsus ning vajadus erinevate osapoolte vahel muutunud möödapääsmatuks. Koostööpartnerite vahelise infovahetuse sujuvaks toimimiseks on tarvis ühtset ja hästi funktsioneerivat lahendust. Käesolev töö keskendub just sellise lahenduse loomisele, et tagada infovahetuse võimalused Eesti Jalgpalli Liidule ning seekaudu toetada jalgpalli jätkuvat päevakohasust Eestis.

Käesoleva bakalaureusetöö eesmärk on luua Eesti Jalgpalli Liidule rakendustarkvara liides, mida on võimalik kasutada kõikidel organisatsiooni koostööpartneritel päringute tegemiseks tarviliku informatsiooni kohta. Tulemusena luuakse ühtne teenus, mis on valmis avalikuks kasutamiseks ja võimaldab lihtsat ning sujuvat andmevahetust.

Bakalaureusetöö on jaotatud peatükkideks. Teises peatükis antakse ülevaade probleemist, mida lahendati lõputöö raames ning organisatsioonis eelnevalt olemasolevatest lahendutest. Kolmandas peatükis kirjeldatakse valminud süsteemi nõudeid ning kirjeldatakse valitud tehnoloogiaid. Neljas peatükk koosneb arenduse protsessi kirjeldusest. Viiendas peatükis esitatakse ja analüüsitakse valminud toodet, kirjeldatakse valideerimise protsessi ja loetletakse edasiarenduse võimalusi.

2 Taust

Selles peatükis kirjeldatakse töö tausta ehk püstitatakse probleem, mida käesolev bakalaureusetöö lahendab ning kirjeldatakse ettevõttes eelnevalt kasutusel olevaid API-laadseid lahendusi.

2.1 Probleemipüstitus

Eesti Jalgpalli Liit on organisatsioon, mis haldab kohalikke jalgpallikoondisi ja Eesti esindusmeeskonda. Liit korraldab samuti kõiki kohalikke jalgpallimeistrivõistlusi, kõige tähtsamaks ja tuntumaks neist on Meistriliiga. Organisatsioon on ka Rahvusvahelise Jalgpalliliidu (FIFA) ning Euroopa Jalgpalliliidu (UEFA) liige [1].

Eestis tegutsevatele jalgpalliklubidele, uudisteportaalidele ja muudele sidusrühmadele on igapäevaseks toimimiseks vajalik pidev ligipääs ajakohasele informatsioonile koondiste, mängude ning muu seotud päevakohase informatsiooni kohta. Eesti Jalgpalli Liit on seega vastutav sellise infovahetuse võimaluse pakkumise eest osapooltele.

Enne käesoleva bakalaureusetöö tegemist ei eksisteerinud Eesti Jalgpalli Liidus ühtset lahendust infovahetuse võimaldamiseks. Organisatsioonis olid kasutusel mitmel erineval tehnoloogial põhinevad API-laadsed süsteemid, aga mitte ükski neist ei olnud terviklik ega kasutatav kõikide osapoolte poolt. Lahenduste varieerumine tekitas probleeme organisatsiooni arendusmeeskonnale, sest mitmeid erisuguseid lahendusi on keerukas hallata ning töötavatel tarkvaraarendajatel ei ole pädevust kõikides kasutatud tehnoloogiates. Lisaks olid kõik süsteemid loodud erinevate arendajate poolt, niisiis olid olemasolevad lahendused täielikult isoleeritud ning ei toetanud standardiseeritud andmevahetust. Kõik eelnimetatu tekitas täiendavaid probleeme sidusrühmade jaoks, kellel oli raskusi ajakohaste ja usaldusväärsete andmete saamisega. Osapooltel esines erinevaid takistusi nagu andmete mittesobivused, kus töötajate ressurss kulus manuaalselt andmete sisestamisele, ning viivitused, mil andmed ei olnud tarvilikul hetkel kättesaadavad.

Probleemi lahendamiseks otsustati luua täiesti uus ning usaldusväärne tarkvaraliides, mida Eesti Jalgpalli Liidu IT-osakond suudaks hallata ning vajadusel edasi arendada. Uus rakendus asendaks töö kirjutamise hetkel eksisteerivad ning kasutusel olevad lahendused ning võimaldaks andmevahetust teostada kõikidel osapooltel, kes seda teha soovivad.

2.2 Olemasolevad lahendused

Järgnevalt esitatakse ning kirjeldatakse kõik andmevahetuslahendused, mis olid eelnevalt Eesti Jalgpalli Liidus kasutusel, analüüsitakse nende eeliseid ning puudujääke ning kirjeldatakse uue rakendusliidese loomise vajadust.

2.2.1 Mobiilirakendus “Eesti jalgpall”

Eesti Jalgpalli Liidul on arendatud mobiilirakendus "Eesti Jalgpall", milles on võimalik jälgida reaalajas võistluseid ja mänge, lugeda uudiseid ning vaadata otseülekandeid ja videoklippe. Mobiilirakenduse taustsüsteemiks on C# keeles ning .NET raamistikus loodud API. Antud liidest kasutavad ka mõned valitud jalgpalliklubid, kes soovivad kindlaid andmeid pärida ja neid omakorda enda kodulehtedel kuvada.

Mobiilirakenduse taustsüsteem on täielikult funktsioneeriv, kuid peamisteks probleemideks liideses on selles kasutusel olevad tehnoloogiad ning üldine disain. Rakendusliides valmis Eesti Jalgpalli Liidule tellimusprojektina, ning see tehti valmis C# keeles ning .NET raamistikus. Organisatsioonis töötavatel IT-osakonna tarkvaraarendajatel puudub aga pädevus antud tehnoloogiates, ehk antud liidest ei ole võimalik hetkeseisuga hallata ega edasi arendada. Kuna rakendusliides loodi esialgu täielikult mobiiliäpi jaoks (arenduses ega selle planeerimise protsessis ei mõeldud muude kasutajate peale), siis liidese vastuste struktuur on samuti mobiilirakenduse järgi tehtud ning üldkasutuse peale ei oldud mõeldud. Antud põhjusel oleks .NET API universaalseks muutmine keerukas ning nõuaks erisuguseid muudatusi olemasolevas koodis. Uue rakendusliidese loomise puhul oleks võimalik teha arendustööd tehnoloogiates, mida on võimalik organisatsiooni töötajatel hiljem hallata ning soovi/vajaduse järgi lisaarendusi luua, ning oleks võimalik ka luua vastused universaalsust ning jätkusuutlikkust silmas pidades.

2.2.2 xml.jalgpall.ee

Eesti Jalgpalli Liidus on kasutuses XML vastustel põhinev lahendus, mis loodi andmeedastuseks Eesti Rahvusringhäälingule, et nad saaksid kuvada koosseise otseülekannetes. Lahenduse raames loodi erinevad failid ning igas failis luuakse ühendus andmebaasiga, teostatakse andmebaasi päring ning vormistatakse XML kujul vastus.

XML lahendus on samuti täielikult funktsioneeriv, kuid ei ole kooskõlas organisatsiooni soovidega andmevahetuseks. Suurim vastuolu antud lahenduse puhul on vastuste formaat - Eesti Jalgpalli Liit soovib vastuseid välja anda JSON kujul, mis on enamlevinud, inimloetavam ning ei sisalda endas ebavajalikku lisainformatsiooni, mis ei oma kliendi vaatest mitte mingisugust tähtsust. Lisaks on problemaatiline lahenduse struktuur - igas failis eraldiseisvalt on olemas iga päringu teostamise samm ehk puudub kihistatud arhitektuuridisain, mis teeb liidese haldamise ning skaleeritavuse keerulisemaks. XML lahendus on tehtud ühe kindla osapoole soovidel põhinedes, ehk vastuste struktuur ei ole universaalne. Kuna lahendus põhineb täielikult XML formaadil ning ei ole loodud olemeid vastuse formaadi tagastamiseks, siis oleks refaktoreerimine ajakulukas töö ning struktuuri puudulikkuse tõttu ei oleks see üldiselt mõistlik.

2.2.3 soccernet.ee

Soccernet on internetiportaal ning üks Eesti Jalgpalli Liidu sidusrühmadest, kus kajastatakse Eesti ning välismaiseid jalgpalliuudiseid, esitatakse erinevaid tulemusi ja statistikaid, postitatakse pilte ja videoid ning avaldatakse meelelahutuslikke- ja arvamusartikleid. Soccernet näitab ka mõningate jalgpallikohtumiste otseülekandeid [2]. Soccernet pärib andmeid Eesti Jalgpalli Liidult ilma liideseta ning loob ühenduse otse andmebaasiga, kust omakorda pärib andmeid.

Otsese andmebaasile juurdepääsu eelis rakendusliidese kasutamise ees on lihtsus – hõlpsalt on võimalik lugeda, kirjutada ning muuta andmebaasi kirjeid ilma rakendusliidese loomiseta, mis nõuab aega ning ressurssi. Sellisest lahendusest võib piisata lihtsamate rakenduste ning kontrollitud keskkondade puhul, kus paindlikkus ei oma suuremat tähtsust [3]. Andmebaasiga otseühenduse implementatsiooni puuduseks on suurem vajadus manuaalse turvahalduse järele, kuna selline lahendus soodustab erinevaid küberturbe riske: SQL süstimine (ingl. k. *SQL injection*), teenustõkestusründed ja (hajutatud) teenustõkestusründed

(ingl. k. *(Distributed) Denial of Service attacks*), kurivara levitamine. Eesti Jalgpalli Liidu puhul on küberturbe aspekt eriti oluline, kuna andmebaasi tabelid sisaldavad tundlikku informatsiooni sidusisikute ja -rühmade kohta ning näiteks nooremate liigade mängijatega seosnev informatsioon ei tohi olla üldse avalikustatud. Rakendusliidese loomine ja kasutamine oleks antud olukorras eelistatum, sest nende raamistikke ja haldusplatvorme uuendatakse regulaarselt turvafunktsioonide ja -parandustega. Lisaks pakuvad need mitmeid erinevaid autentimisprotokolle juurdepääsu haldamiseks ning krüpteerivad kliendi ja serveri vahelisi andmeid sideprotokollide abil, mis aitab turvahaldust automatiseerida ning annab rakendusliidese kasutamisele suure eelise andmebaasiga otseühenduse ees [4].

2.3 Uue liidese loomine

Kõikide olemasolevate lahenduste analüüsimisest selgus, et kõige mõistlikum variant oleks luua täiesti uus rakendusliides, mille puhul eksisteerib Eesti Jalgpalli Liidus töötavatel tarkvaraarendajatel pädevus tehnoloogiates, vastused on eelistatumas formaadis ning selgemini vormistatud, projekti arhitektuur ja disain on läbimõeldud, selged ja jätkusuutlikud ning et lahendus oleks piisavalt üldine, kuid ammendav, et seda oleks võimalik kasutada kõikidel osapooltel, kes seda vähegi peaksid soovima teha.

3 Lahenduse analüüs

Selles peatükis esitatakse arendatud rakenduse funktsionaalsed ning mittefunktsionaalsed nõuded, loetletakse lõputöös kasutatud tehnoloogiad ja nende alternatiivid ning selgitatakse tehtud valiku langetamise põhjuseid.

3.1 Süsteemi funktsionaalsed nõuded

Järgnevalt loetletakse lõputöö raames loodava rakendustarkvara liidese funktsionaalsed nõuded.

- Süsteemi rakendusliides koosneb lõpp-punktidest, mis võimaldavad klientidel teostada CRUD operatsioonidest R ehk lugemise (ingl. k. *Read*) operatsiooni (välja arvatud märgipõhise autentimise lõpp-punktid).
- Süsteem tagastab kõik vastused inimloetavas vormingus. Erinevad lõpp-punktid, mis tagastavad sama tüüpi olemeid, on ühtse struktuuriga.
- Süsteem rakendab serveripoolset vahemällu salvestamist. Süsteem kontrollib enne andmebaasi päringute teostamist vahemälu ja tagastab vahemällu salvestatud andmed, kui need on saadaval.
- Süsteem pakub veakäsitlust ning kõik sissetulevad päringud valideeritakse serveri poolel. Liides tagastab informatiivseid veateateid, kui peaks juhtuma kliendi viga (puudulik või kehtetu sisend) või serveri tõrge. Vastuses tagastatakse inimloetavas vormingus HTTP olekukood ning informatiivne sõnum.
- Süsteemis eksisteerib märgipõhine autentimine. Märke hoitakse Eesti Jalgpalli Liidu andmebaasis krüpteeritud kujul. Kaitstud lõpp-punktid nõuavad autentimist Laraveli sisseehitatud autentimismeetodite kaudu.
- Süsteem toetab loendi tüüpi tagastavate lõpp-punktide korral vastuste pagineerimist. Vastus kajastab vastuse sisule lisaks ka praegust lehekülge ning lehe suurust.

3.2 Süsteemi mittefunktsionaalsed nõuded

Järgnevalt loetletakse lõputöö raames loodava rakendustarkvara liidese mittefunktsionaalsed nõuded.

- Uue liidese tehnoloogiate valimisel arvestatakse nende sobivusega Eesti Jalgpalli Liidu ökosüsteemi ning IT-osakonna pädevustega, et tagada lahenduse jätkusuutlikkus, hõlbus edasiarendus ja haldus organisatsiooni arendajate poolt.
- Süsteem tagastab vastuseid mõistliku aja piires - võimalikult kiirelt, kuid mitte rohkem, kui 2 sekundi jooksul [5].
- Süsteem teostab krüpteeritud andmevahetust, kasutades HTTPS-i ning vajadusel märgipõhist autentimist.
- Süsteem on modulaarne, kus kontrollerite, teenuste, repositooriumite ja ruuteri ülesanded on selgelt eraldatud. Kood järgib Laraveli paremaid tavasid ja printsiipe, et edaspidist hooldust lihtsustada.
- Süsteemile luuakse integratsioonitestid, et oleks võimalik veenduda süsteemi töökindluses.
- Süsteemile tehakse manuaalsed jõudlustestid, et tagada rakenduse stabiilne toimimine *live* keskkonnas.
- Rakendusliidesel on selge ja ammendav dokumentatsioon, kus on kajastatud kõik lõpp-punktid ja nende eesmärgid, päringu parameetrid ja vastuse vormingud.

3.3 Rakendusliidese arhitektuur

Rakendusliidese arhitektuuri valik on tarkvaraarenduses oluline otsus, mis mõjutab rakenduse jõudlust, laiendatavust ja kasutajakogemust. Kaasaegsetes veebiteenustes on enam levinud kuus lahendust: REST, SOAP, GraphQL, gRPC, WebSocket ja *webhook*'id. Kuigi kõik eelnimetatud tehnoloogiad võimaldavad kliendi ja serveri vahelist andmevahetust, erinevad need oma struktuuri, paindlikkuse ja kasutusjuhtude poolest. Kuna muud tehnoloogiad ei vasta täielikult Eesti Jalgpalli Liidu vajadustele, analüüsitakse järgnevalt kahte potentsiaalselt sobivaimat lahendust: REST ja GraphQL.

3.3.1 REST

REST (*Representational State Transfer*) on arhitektuuriline stiil, mida kasutatakse laialdaselt veebiteenuste loomisel. REST põhineb HTTP-protokollil ja järgib kindlaid põhimõtteid nagu ressursside selge adresseerimine URL-ide kaudu, staatuse hoidmine serveri poolel ning operatsioonide ja andmevahetuse määratlemine standardsete HTTP-meetoditega (GET, POST, PUT, DELETE).

REST põhineb viiel juhtimisprintsiibil, mis seavad RESTilikule arhitektuurile piirangud:

1. REST määratleb järjepideva ja ühtse liidese klientide ja serverite vaheliseks suhtluseks. Ressursse tuvastatakse, kasutades standardseid HTTP-meetodeid (GET, POST, PUT, DELETE) ning URI-sid (Uniform Resource Identifiers).
2. REST järgib klient-server disainimustrit ehk kliendi ja serveri komponendid on suhtluses eraldiseisvad. Antud omadus aitab liideste skaleeritavust tõsta.
3. REST on olekuvaba ehk serveril puudub päringuväline kontekst. Kliendi poolt tehtud päring peab sisaldama endas kogu vajalikku informatsiooni, et seda oleks võimalik täita.
4. RESTil on sisseehitatud vahemällu salvestamise võimekus, ning iga vastus peab omama märgistust vahemällu salvestamise vajaduse kohta.
5. Kihiline süsteemistiil võimaldab murede selge eristamise.

REST-i peamised eelised on lihtsus nii arendajale kui kliendile, ning laialdane tugi erinevates programmeerimiskeeltes ja raamistikus. Kuna REST API-d kasutavad tekstipõhiseid formaate nagu JSON ja XML, on need hästi loetavad ja hõlpsasti integreeritavad teiste süsteemidega. Samuti võimaldab REST-i struktuur paremat ning lihtsat vahemällu salvestamist ja jõudluse optimeerimist.

Siiski on REST-il ka piiranguid. Kuna iga päring tagastab kindlaksmääratud andmestruktuuri, võib see põhjustada liigsete andmete edastamist (*over-fetching*) või ebapiisavat andmete edastamist (*under-fetching*). Viimasega võivad tekkida olukorrad, kus kliendil on tarvis pöörduda lõpuks mitme erineva päringuga serveri poole, sest ei saa piisavalt informatsiooni kätte (ning selle tagajärjel võib ka liigse, ebavajaliku info kätte saada). Seetõttu võib keerukamate rakenduste puhul REST-i jäik arhitektuur muutuda koormavaks, kui erinevad kliendid vajavad erisuguseid andmeid enda toimetuste jaoks [6, 7].

3.3.2 GraphQL

GraphQL on Facebooki poolt välja töötatud päringukeel ja andmevahetusstandard. API arhitektuuri unikaalsus alternatiivide ees seisneb võimaluses klientidel määratleda täpselt, millist teavet (milliseid välju) on tarvis serverilt pärida. Antud omadusega parandab GraphQL suure probleemi, mis eksisteerib teiste arhitektuuride puhul - liigne andmeedastus. Selle asemel, et kasutada fikseeritud lõpp-punkte, pärib klient andmeid ühe dünaamilise POST päringu kaudu, mis võib hõlmata mitut seotud andmeobjekti. GraphQL pakub ka tellimuspaketti, mis võimaldab saada reaalajas värskendusi, kui andmed muutuvad. Arhitektuuristiili kasutavad mitmed mainekad ettevõtted nagu Github ja Shopify.

GraphQL-i peamine eelis on paindlikkus, kuna see võimaldab klientidel küsida täpselt vajaminevat teavet ning kombineerida mitmest erinevast allikast pärinevaid andmeid ühte päringusse. Antud võimekus muudab API efektiivsemaks ja vähendab võrgu koormust. GraphQL sobib seetõttu suurepäraselt suurte ja keeruliste andmetega töötavatele aplikatsioonidele. Lisaks võimaldab GraphQL tugevat tüübisüsteemi, mis aitab ennetada vigu ja lihtsustab ka arendaja tööd.

GraphQL-i rakendamine võib võrreldes RESTiga olla palju keerukam, kuna see nõuab täiendavat serveripoolset töötlemist. Arhitektuuristiil on tuntuselt ja kasutuselt palju madalam RESTist, ning ka vähem kasutajasõbralik - REST võimaldab kõigest URL-i sissekirjutamisel brauserisse vastus tagastada, kuid GraphQL vajab päringu teostamiseks eraldi tarkvara seadistamist. GraphQL'il puudub ka sisseehitatud vahemällu salvestamise võimekus ning selle implementeerimine nõuab täiendavate teekide kasutamist, mis võib teha selle implementeerimise keerukaks [6, 8].

3.3.3 Arhitektuuri valik

Antud bakalaureusetöö rakendusliidese jaoks kasutatakse REST arhitektuuristiili. Otsustuspunktideks said laialdane kasutus ning kättesaadavus, vahemälu võimekus ning selle üldine olemasolu. Antud rakenduse puhul ei põhjusta kindlaksmääratud andmestruktuur suuremaid probleeme liigse andmeedastusega, sest sidusrühmad soovivad valdavalt Eesti Jalgpalli Liidult pärida sarnase struktuuriga informatsiooni - kui päritakse infot mängu kohta, soovitakse ka infot meeskondade, koosseisude, sündmuste jms kohta.

3.4 Programmerimiskeel ja raamistik

REST API populaarsus on käesoleva töö kirjutamise ajal endiselt kasvutrendis, niisiis peab API arendamisel tegema optimaalse programmeerimiskeele valiku, et liides oleks jätkusuutlik. Valiku langetamisel tuleb arvestada arendusmeeskonna oskustega tehnoloogiate osas, süsteemi funktsionaalsete nõuete ning üldiste eesmärkidega, mida liides täitma hakkab ning mida selle saavutamiseks vaja teha on. Igal programmeerimiskeelel on erinev jõudlus vastavalt sellele, kui optimeeritud antud keel on. Samuti on oluline, et keel pakub tugevaid teeke, raamistikke ja süntaksit ning keelt ning selle raamistikke uuendatakse ning hoitakse tänapäevasena loojate poolt. Keel peaks võimaldama ka horisontaalset skaleerimist ilma suuremate ümberkirjutusteta, kuna antud omadus on API kasvu puhul kriitilise väärtusega [9, 10].

REST API-de arendamisel on enim arendajate poolt eelistatud keeled JavaScript, Python, Java, PHP, C# ja Go [9]. Käesoleva rakendusliidese arendamisel kasutatakse PHP keelt ning Laraveli raamistikku.

3.4.1 PHP. Laravel

PHP (*Hypertext Preprocessor*) on laialdaselt kasutatav serveripoolne skriptikeel, mida kasutatakse enim veebiarendusel. Keel on maailma suurima ajaveebisüsteemi WordPress keskmes, ning seda kasutavad veel suured ettevõtted nagu Facebook, Wikipedia ja Shopify. 2025. aasta aprilli seisuga on PHP Tiobe indeksi järgi programmeerimiskeelte seas populaarsuselt 13. kohal tõusva hinnanguga. PHP on pälvinud enda populaarsuse tänu platvormist sõltumatusele, avatud lähtekoodile, lihtsale õppimiskeerukusele, pea kõikide andmebaasidega sünkroniseerimisele ja suurele kogukonnale. PHP raamistikud nagu Laravel on sobilikud robustsete ning efektiivsete API-de loomisel [11, 12, 13].

3.4.2 Programmerimiskeele ja raamistiku valik

Käesoleva lõputöö raames luuakse rakendusliides PHP keeles ning Laraveli raamistikus. Valiku langetamisel olid tähtsamateks otsustuspunktideks autori isiklik huvi tehnoloogiate vastu, tehnoloogiate sobilikkus liidese loomiseks ning Eesti Jalgpalli Liidu IT-osakonna arendajate pädevus keele ning raamistikuga. Antud tehnoloogiate valik on ka seetõttu eelistatud, et Eesti Jalgpalli Liidu kodulehe *front-end* on samuti Laravelis kirjutatud. Niisiis

on liidese samakeelseks tegemisel hõlpsam ühist projekti hallata, sest kaob eraldiseisvate repositooriumite vajadus. Samuti teeb ühine repositoorium võimalikuks autentimise ning äri- jms. loogikate jagamise eesrakenduse ja API vahel, mis kaotab koodi duplikeerimise vajaduse [14].

3.5 Andmevahetusvorming

REST API-des on enamlevinud andmevahetusvormingud XML ning JSON. XML on samaaegselt keel ning andmevorming, mis võimaldab erinevaid funktsioone andmevorminguks ning selle eelistamine on põhjendatud, kui andmete oluline osa on metaandmed ning dokumendi märgistus. JSON-i eesmärk on teostada struktureeritud andmevahetust, esindades erinevaid andmetüüpe ja objekte ilma metaandmete ja dokumendi märgistusteta. Üks JSON-i eelis on serialiseerimise ning deserialiseerimise kiirus võrreldes XML-iga, mida võimaldab JSON-i väiksem suurus sama informatsiooni hulga puhul [15].

Antud bakalaureusetöö raames loodud liideses kasutatakse andmete tagastamise vormindamiseks JSON formaati. Antud formaat on töö kirjutamise ajal palju enamlevinud, ning kuigi kumbki vorming pole teisest absoluutselt parem, siis JSON on käsitletavas rakendusliideses sobilikum seetõttu, et Eesti Jalgpalli Liidu kontekstis ei oma metaandmete ning dokumendi märgistuse saatmine kliendile tähtsust.

3.6 Vahemälu

Eesti Jalgpalli Liidule tehakse kindlatel perioodidel päringuid sarnaste andmete kohta; näiteks uue mängu toimumise päeval võib organisatsioonile saabuda kümneid korduvaid päringuid sama mängu tulemuste või mängueelse info kohta. Selleks, et rakendusliidese vastamise kiirust hoida optimaalsena, on tarvis andmebaasi koormust vähendada vahemälu võimekuse abil. Käesoleva töö raames kasutatakse vahemälu tehnoloogiana Redist (*Remote Dictionary Server*), mis on mälusisene võtme-väärtuse põhine andmebaas. Tehnoloogiat kasutatakse peamiselt kiirreageerivate andmebaaside tehnoloogiana või rakenduste vahemäluna. Redise suurem eelis ülejäänud NoSQL andmebaaside ees on andmete salvestamine otse operatiivmälus, mitte kettaseadmetel. Mainitud omadus teeb Redise kiireks töökindlaks ja tagab jõudluse [16]. Tehnoloogia sobib antud töö kontekstis ka seetõttu, et organisatsiooni arendusmeeskonnal on pädevus selle kasutamisel.

4 Arendus

Selles peatükis kirjeldatakse töö arenduskäiku ehk selgitatakse liidese valmimise protsessi ning esitatakse kõik implementeeritud funktsionaalsused.

4.1 Eeltöö

Enne rakenduse arendamisega alustamist toimusid mitmed kohtumised juhendajatega, mille eesmärgiks oli kooskõlastada tehnoloogiate valikud, määratleda süsteemi kasutajate vajadused ning arutada, kuidas kõikide osapoolte erinevaid huvisid arvestades luua kasutajasõbralik ja funktsionaalne liides. Kõikide klientide vajadustega arvestamiseks analüüsiti ning ühildati olemasolevate lahenduste funktsionaalsused. Täpsemad arhitektuurilised otsused teostas töö autor iseseisvalt – kuidas koodi struktureerida, millised olemid luua jne.

Rakendusliidese arendamiseks oli töö autoril vajalik analüüsida ka Eesti Jalgpalli Liidu andmebaasi, mille peale liides üles ehitati. Organisatsiooni andmebaasil ei eksisteerinud liidese valmimise ajal ammendavat dokumentatsiooni ning autoril puudus konkreetse andmebaasiga ka varasem kokkupuude, seega kujunes selle struktuuri ja loogika mõistmine ajamahukaks ja põhjalikku analüüsi eeldavaks protsessiks.

Koosolekute ja isiklike eeltööde tulemusena koostati esialgne arhitektuuriline ülevaade, milles määratleti lõpp-punktides tagastatavad olemid, seosed nende vahel ning REST API lõpp-punktid. Eeltöö aitas kaasa arendusetappide sihipäraseks kavandamiseks ja lõi aluse selgepiiriliseks ülesannete jaotuseks iteratiivses arendusprotsessis.

4.2 Arendusvõtted

Rakenduse arendamisel kasutati agiilse arenduse põhimõtteid, kus töö jaotati väiksemateks kahenädalasteks iteratsioonideks ehk sprintideks. Iga sprindi alguses püstitati üldine eesmärk, mida sooviti järgneva kahe nädala jooksul valmis teha. Iga sprindi lõpus analüüsiti antud sprindis püstitatud eesmärgi täitmist ning tehti koosolek koos bakalaureusetöö

koolipoolse juhendajaga. Koosolekus esitles autor vahepealsel perioodil tehtud tööd, esitas juhendajale tekkinud küsimusi ning juhendaja tegi ka endapoolsed ettepanekud, kui tunnetas selle vajadust. Edusamme esitleti paralleelselt ka ettevõttepoolsele juhendajale. Töö autoril oli ka võimalus mõlema juhendajaga suhelda igapäevaselt sõnumite kaudu, kui tekkis jooksvaid küsimusi sprintide jooksul.

Arendamise käigus kasutati järgnevalt loetletud praktikaid:

- kogu lähtekoodi hallati Git kasutades ning valminud projekt asub GitLabis. Iga alamülesande jaoks loodi GitLabis sprindiga seotud *issue*, millele lisati juurde grupeerivad sildid (ingl. k. *label*). Siltide järgi oli lihtne alamülesande eesmärki ja tüüpi eristada ning filtreerida;
- koodi kirjutamisel järgiti Laraveli paremaid tavasid, et saavutada paremini hallatav, laiendatav ja testitav kood;
- kõik lõpp-punktid järgivad REST API standardeid, järgides selgeid URLi struktuure;
- arendusel kasutati JetBrains'i integreeritud programmeerimiskeskkonda PHPStorm ning andmebaasi haldamiseks phpMyAdmin'i;
- API spetsifikatsioon loodi OpenAPI standardi järgi ning dokumentatsioon genereeriti automaatselt.

4.3 Olemid

Rakendusliideses on olemas erinevad olemid, mis on omavahel seotud (vt Joonis 1 ja Joonis 7). Lõpp-punktid tagastavad järgnevalt loetletud olemeid:

- Klubi (*Club*) - esindab ühte jalgpalliklubi. Objekt sisaldab endas nime, logo, esimeest ja aktiivsust (kas klubi on päringu tegemise hetkel tegutsev).
- Võistlus (*Competition*) - esindab ühte jalgpallivõistlust. Objekt sisaldab endas nime, võistlustabelit, kategooriat (muru-, saali-, rannajalgpall jne.) ning aktiivsust (kas võistlus on päringu tegemise hetkel tegutsev).
- Mäng (*Game*) - esindab ühte jalgpallimängu. Objekt sisaldab endas mängu toimumiskohta ning -aega, võistlusesse kuuluvust, osalevaid kohtunikke, osalevaid meeskondi ja nende koosseise (mängus osalevaid mängijaid), mängus toimunud sündmusi.
- Mängija (*Player*) - esindab ühte jalgpallimängijat. Objekt sisaldab endas ees- ja

perekonnanime, särginumbrit, mängija positsiooni, portreefotode URL aadresse, kodakondsust, sünniaega, aktiivsust (kas mängija on päringu tegemise hetkel tegutsev) ning praegust meeskonda kuuluvust (juhul, kui mängija on aktiivne). Kui mängijat kuvatakse mängu objekti sees, siis kuvatakse ka seda, kas mängija on mängu algkoosseisus ning kas mängija on meeskonna kapten.

- Statistika (*Statistic*) - esindab ühte statistikat. Statistika võib käia kas konkreetse mängija, meeskonna või võistluse kohta. Objekt sisaldab endas nii kokku arvutatud kui keskmisi arve mängitud mängude, mängitud minutite, väravate, väravasöötude, punaste ja kollaste kaartide kohta. Võistluste ja meeskondade statistikates kuvatakse infot ka pealtvaajatate arvu ning enim väravaid löönud mängija kohta.
- Meeskond (*Team*) - esindab ühte jalgpallimeeskonda. Objekt sisaldab endas meeskonna nime, logo, koduväljakut, treenereid, mängijate koosseisu, klubilist kuuluvust ning aktiivsust (kas meeskond on päringu tegemise hetkel tegutsev). Kui meeskonda kuvatakse mängu objekti all, siis kuvatakse ka meeskonna sees mänguvormi.

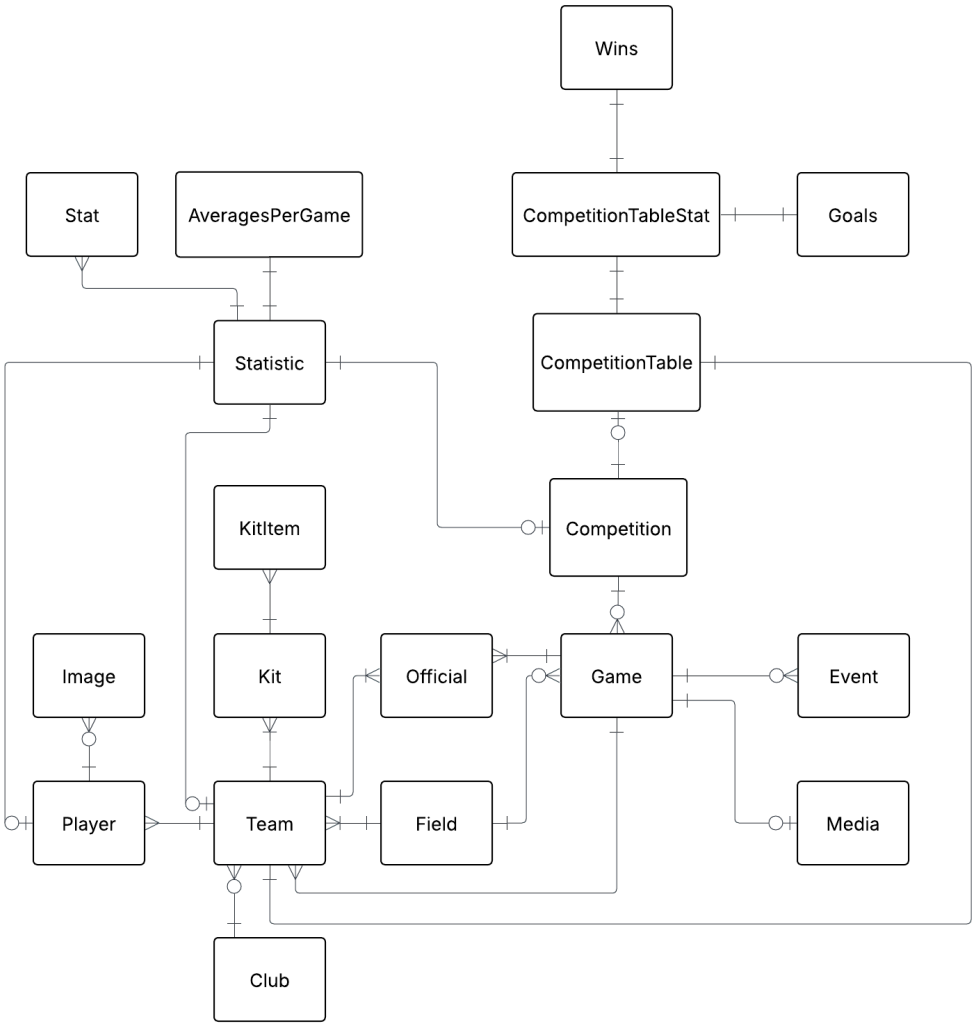
Järgnevalt loetletakse olemid, mida ei tagastata lõpp-punktides, kuid sisalduvad vastustes tagastatavate objektide sees:

- Sündmus (*Event*) - esindab ühte sündmust, mis toimus konkreetses jalgpallimängus. Sündmus võib olla värav, penalti, kollane kaart, punane kaart või mängijate vahetus. Objekt sisaldab endas informatsiooni sündmuse tüübi ning seosneva(te) isiku(te) kohta.
- Väljak (*Field*) - esindab ühte jalgpalliväljakut. Objekt sisaldab endas nime ning asukohta (täpne aadress või riik).
- Ametnik (*Official*) - esindab ühte jalgpalliametnikku. Olem võib esindada nii meeskonnatreenerit kui mängukohtunikku. Objekt sisaldab endas ees- ja perekonnanime ning rolli. Roll võib olla kohtuniku puhul kas väljakukohtunik, 2.-4. kohtunik, 1.-2. lisakohtunik, inspektor või vaatleja. Meeskonnatreeneri puhul võib roll olla kas peatreener või 1.-2. abitreener.
- Mänguvorm (*Kit*) - esindab jalgpallimeeskonna mänguvormi. Mänguvormi kuvatakse mängu objekti sees mängivate meeskondade all. Objekt sisaldab endas vormi tüüpi (kodu-, võõrsil- või kolmas vorm), mängija tüüpi (väravavaht või väljakumängija) ning järjendit vormiriietest.

- Mänguvormi osa (*KitItem*) - esindab jalgpallimeeskonna mänguvormi osa. Mänguvormi osi kuvatakse mänguvormi objekti sees. Objekt sisaldab endas vormiriietuse osa (särk, sokid, püksid), primaarvärvi ja sekundaarvärvi (*hex* koodid) ning värvivariatsiooni (ühevärviline, horisontaal- või vertikaaltriibud).
- Võistlustabel (*CompetitionTable*) - esindab ühe jalgpallivõistluse liigatabeli rida kindlal hooajal. Tabel näitab meeskondade järjestust ning nende mõningaid statistilisi näitajaid kindlal hooajal antud võistluses. Objekt sisaldab endas järjestuse kohta tabelis, meeskonda ning tabeli statistikat.
- Võistlustabeli statistika (*CompetitionTableStat*) - esindab ühe jalgpallivõistluse liigatabeli meeskonna statistikat kindlal hooajal. Objekt sisaldab endas meeskonna kogutud punkte, mängitud mängude arvu, võite, kaotuseid, viike ning väravaid.
- Võidud (*Wins*) - esindab ühe jalgpallivõistluse liigatabeli meeskonna võite kindlal hooajal. Objekt sisaldab endas arvuliselt kokku saadud võite, võite lisaajal, võite penaltitega ning võite võõrsil mängides.
- Väravad (*Goals*) - esindab ühe jalgpallivõistluse liigatabeli meeskonna väravaid kindlal hooajal. Objekt sisaldab endas arvuliselt kokku löödud väravaid, vastaste löödud väravaid, enda ja vastaste löödud väravate vahe ning kodus ja võõrsil löödud väravate arvu.
- Alamstatistika (*Stat*) - esindab ühte statistika alamjaotust. Objekt käib konkreetse statistilise näitaja kohta (nt mängitud minutid või väravad). Objekt sisaldab endas näitaja koguväärtust ning lisaks näitajat jaotatuna karika- ja muude võistluste vahel (nt minutite puhul kokku mängitud minutid, karikamängudes mängitud minutid ning muud mängitud minutid).
- Keskmised mängu kohta (*AveragesPerGame*) - esindab statistika olemi sees näitajate keskmisi väärtusi kas mängija, võistluse või meeskonna kohta. Objekt sisaldab endas keskmisi väravaid, väravasööte, kollaseid ja punaseid kaarte, vaatajaid ja mängu kestvust ühe mängu kohta.
- Roll (*Role*) - üldine klass, mis esindab erinevaid olemeid Eesti Jalgpalli Liidu andmebaasis. Roll esindab niisuguseid objekte, millel on olemas täpselt kolm atribuuti - identifikaator, nimi ja kood. Roll on kasutusel näiteks mänguvormi tüübi, mängija positsiooni, eelnimetatud ametniku rolli jm. puhul.
- Meedia (*Media*) - esindab meediat, kes edastab konkreetset mängu. Objekt sisaldab

endas unikaalset identifikaatorit ja nime.

- Pilt (*Image*) - esindab pilti mängijast. Objekt sisaldab endas pildi tüüpi ning aadressi, kust pildi kätte saab.



Joonis 1. *Olemi suhtediagramm.*

4.4 Loodud lõpp-punktid

Järgnevalt loetletakse kõik loodud kuusteist lõpp-punkti, mis on rakenduses saadaval ning nende parameetrite selgitused. Lõpp-punktid on jaotatud gruppidesse vastavalt sellele, millist olemit lõpp-punktid tagastavad (vt Joonis 2).

- GET Games – GET tüüpi lõpp-punktid, mis tagastavad kodu- ja rahvusvahelisi mänge
    1. `games` - tagastab kõik mängud; võimalik filtreerida aja ja tüübi kaudu
        - Päringuparameetrid `year` - aasta, `month` - kuu, `day` - päev, `type` - mängu tüüp: kodumäng või rahvusvaheline mäng
    2. `games/{gameId}` - tagastab konkreetse mängu
        - Aadressiparameeter `gameId` - mängu unikaalne identifikaator
    3. `teams/{teamId}/games` - tagastab mängud, kus konkreetne meeskond on osalev
        - Aadressiparameeter `teamId` - meeskonna unikaalne identifikaator, päringuparameetrid `year` - aasta, `type` - mängu tüüp: kodumäng või rahvusvaheline mäng
    4. `players/{playerId}/games` - tagastab mängud, kus konkreetne mängija on osalev
        - Aadressiparameeter `stakeholderId` - mängija unikaalne identifikaator, päringuparameetrid `year` - aasta, `type` - mängu tüüp: kodumäng või rahvusvaheline mäng
    5. `fields/{fieldId}/games` - tagastab mängud, mis toimusid konkreetsel väljakul
        - Aadressiparameeter `fieldId` - väljaku unikaalne identifikaator, päringuparameetrid `year` - aasta, `type` - mängu tüüp: kodumäng või rahvusvaheline mäng
    6. `competitions/{competitionId}/games` - tagastab mängud, mis toimusid konkreetse võistluse raames
        - Aadressiparameeter `stakeholderId` - mängija unikaalne identifikaator, päringuparameetrid `year` - aasta, `type` - mängu tüüp: kodumäng või rahvusvaheline mäng, `isUpcoming` - tõeväärtus: tulevased/toimunud mängud
- GET Teams – GET tüüpi lõpp-punktid, mis tagastavad kodu- ja rahvusvaheliste meeskondade olemeid
    7. `teams` - üldine lõpp-punkt; tagastab üldise informatsiooni kõikide meeskondade kohta – unikaalne identifikaator ja nimi

- Päringuparameetrid `type` - meeskonna tüüp: kodukoondis või rahvusvaheline meeskond, `name` - nimi, mille vastu meeskonda otsida, `club` - klubi unikaalne identifikaator, mille sekka kuuluvaid meeskondi otsida

8. `teams/{teamId}` - tagastab täpsema informatsiooni konkreetse meeskonna kohta koos koosseisuga
    - Aadressiparameeter `teamId` - meeskonna unikaalne identifikaator, päringuparameetrid `year` - aasta; kui parameeter lisatakse, näidatakse meeskonna koosseisu just sellel aastal; `type` - meeskonna tüüp: kodukoondis või rahvusvaheline meeskond

- GET Competitions – GET tüüpi lõpp-punktid, mis tagastavad kodu- ja rahvusvaheliste võistluste olemeid

9. `competitions` - üldine lõpp-punkt; tagastab üldise informatsiooni kõikide võistluste kohta – unikaalne identifikaator ja nimi
    - Päringuparameetrid `type` - võistluse tüüp: koduvõistlus või rahvusvaheline võistlus, `name` - nimi, mille vastu võistlusi otsida, `isActive` - filtreerimiseks võistluse aktiivsuse järgi, `discipline` - filtreerimiseks võistluse kategooria järgi.
10. `competitions/{competitionId}` - tagastab täpsema informatsiooni konkreetse võistluse kohta koos võistlustabeliga
    - Aadressiparameeter `competitionId` - võistluse unikaalne identifikaator, päringuparameetrid `type` - võistluse tüüp: koduvõistlus või rahvusvaheline võistlus, `year` - aasta; kui parameeter lisatakse, näidatakse just selle aasta võistlustabelit.

- GET Statistics – GET tüüpi lõpp-punktid, mis tagastavad statistikaid erinevate olemite kohta

11. `competitions/{competitionId}/statistics` - tagastab statistilisi näitajaid konkreetse võistluse kohta
    - Aadressiparameeter `competitionId` - võistluse unikaalne identifikaator, päringuparameetrid `type` - võistluse tüüp: koduvõistlus või rahvusvaheline võistlus, `year` - aasta; kui parameeter lisatakse, näidatakse just selle aasta statistikat.
12. `players/{playerId}/statistics` - tagastab statistilisi näitajaid konkreet-

se mängija kohta

  - Aadressiparameeter `playerId` - mängija unikaalne identifikaator, päringuparameetrid `year` - aasta; kui parameeter lisatakse, näidatakse just selle aasta statistikat, `type` - mängija tüüp: kodu- või rahvusvaheline mängija

13. `teams/{teamId}/statistics` - tagastab statistilisi näitajaid konkreetse meeskonna kohta

  - Aadressiparameeter `teamId` - meeskonna unikaalne identifikaator, päringuparameetrid `year` - aasta; kui parameeter lisatakse, näidatakse just selle aasta statistikat, `type` - meeskonna tüüp: kodukoondis või rahvusvaheline meeskond

- GET Players – GET tüüpi lõpp-punktid, mis tagastavad kodu- ja rahvusvaheliste mängijate olemeid

14. `players` - üldine lõpp-punkt; tagastab üldise informatsiooni kõikide mängijate kohta – unikaalne identifikaator ja nimi

  - Päringuparameetrid `type` - mängija tüüp: kodu- või rahvusvaheline mängija, `forename` - eesnimi, mille vastu mängijaid otsida, `surname` - perekonnanimi, mille vastu mängijaid otsida.

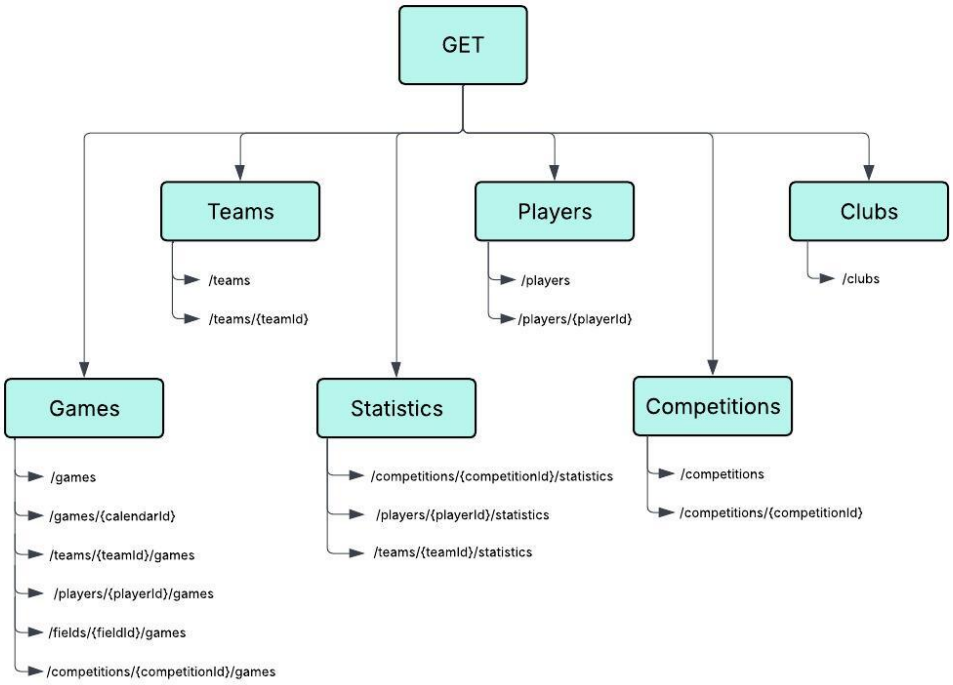
15. `players/{stakeholderId}` - tagastab täpsema informatsiooni konkreetse mängija kohta

  - Aadressiparameeter `playerId` - mängija unikaalne identifikaator, päringuparameeter `type` - mängija tüüp: kodu- või rahvusvaheline mängija.

- GET Clubs – GET tüüpi lõpp-punktid, mis tagastavad jalgpalliklubide olemeid

16. `clubs` - üldine lõpp-punkt; tagastab üldise informatsiooni kõikide klubide kohta – unikaalne identifikaator ja nimi

  - Päringuparameetrid `name` - nimi, mille vastu klubisid otsida, `isActive` - filtreerimiseks klubi aktiivsuse järgi.



Joonis 2. *Rakendusliidese struktuuridiagramm.*

4.5 Lahenduse struktuur

Käesoleva bakalaureusetöö raames arendatud rakenduse valmimisel järgiti domeenikeskse disaini põhimõtteid. Põhimõtete järgimise eesmärk on jagada süsteem loogilisteks kihtideks vastavalt äriloogikale ja tehnilisele teostusele. Selline lähenemine toetab suuremahuliste süsteemide skaleerimist, hooldatavust ning arendajate ja valdkonna ekspertide omavahelist arusaamist.

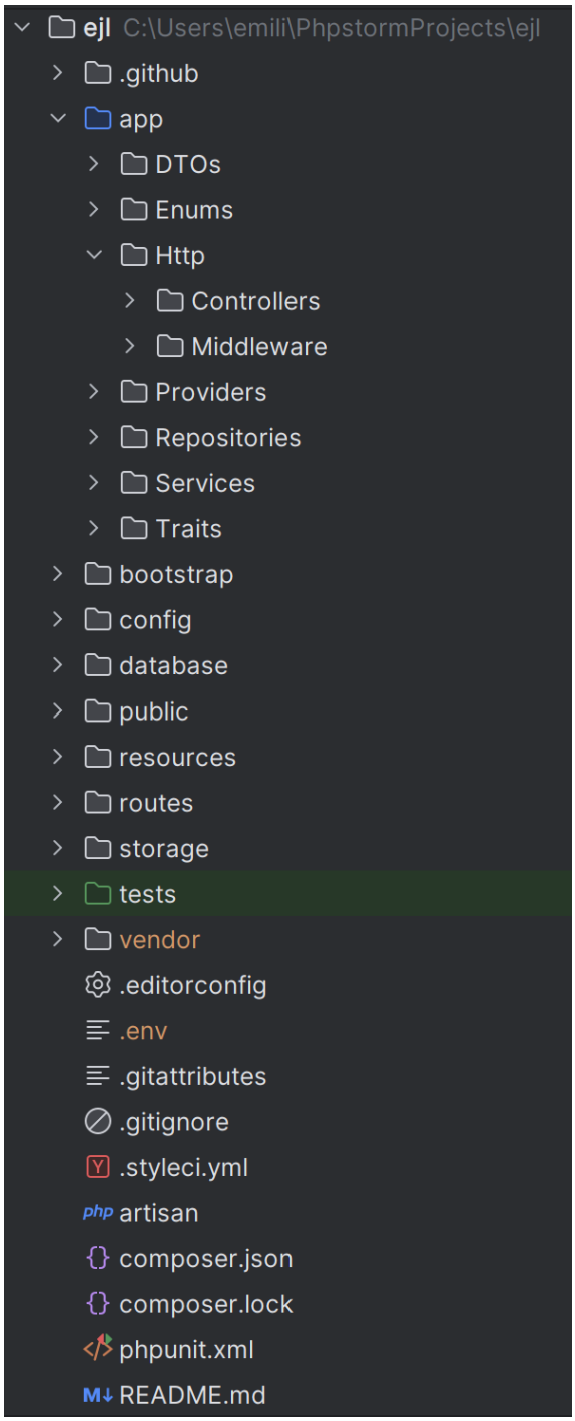
Rakendusliidese äriloogika osa asub projekti `app` kataloogis (vt Joonis 3). Kataloog on omakorda jaotatud järgmisteks peamisteks komponentideks:

- `DTOs/` – sisaldab kõiki olemeid, kuhu rakendus sisestab andmed ning mis vormistatakse liidese vastusteks.
- `Enums/` – enum-tüüpi klassid, mis kirjeldavad rakenduses kasutatavaid staatilisi väärtusi (nt mänguvormi värvivariatsiooni tüüp, mängija pildi tüüp).
- `Http/` – sisaldab rakendusliidesega otseselt seotud komponente. Kataloog jaguneb:
  - `Controllers/` – sisaldab kõiki kontrollereid, mis haldavad päringute ja vastuste voogu ning on ühendused kasutaja ja rakenduse äriloogika vahel.

    - `Middleware/` – kiht, mis vastutab päringute töötlemise eest (nt JSON vormingus vastuste tagastamine).
- `Providers/` – Laravelile omased teenusepakkujad, millega registreeritakse sõltuvused või konfigureeritakse süsteemikäitumist.
- `Repositories/` – andmekiht; vastutab andmebaasiga suhtlemise eest.
- `Services/` – Äriloogika kiht. Loogika rakendamine toimub teenuste klassides, mis võivad kasutada kõiki muid ressursse.
- `Traits/` – sisaldab korduvkasutatavad koodiplokke, mida saab kasutada muudes klassides funktsionaalsuse jagamiseks (nt. organisatsiooni URL aadresside algused, ajavahemike vormindamine).

- `config/, database/, public/, resources/` – vastutavad rakenduse konfiguratsiooni, andmebaasimigraatsioonide, staatiliste failide ja ressursside haldamise eest.
- `routes/` – kaust, kus defineeritakse rakenduse API lõpp-punktid ning seotakse need vastavate kontrolleritega.
- `tests/` – sisaldab automaatteste, mis võimaldavad rakenduse korrektset toimimist kontrollida.
- `vendor/` – välised sõltuvused, mida ei muudeta käsitsi.

Projektis kasutatav arhitektuur võimaldab hoida domeeniloogika tehnilisest kihist eraldatuna, toetades sellega loetavat ja laiendatavat koodibaasi (nt `Services` ja `Repositories` kataloogid peegeldavad tüüpilist domeenipõhist struktuuri, kus äriline loogika ja andmekiht on teineteisest eraldatud).



Joonis 3. *Projekti struktuur.*

4.6 Rakenduse optimeerimine

Süsteemi jõudluse parandamiseks ja koormuse vähendamiseks andmebaasile on rakendatud mitmeid optimeerimisvõtteid. Käesolevas peatükis käsitletakse kahte peamist lähenemist: vahemälu kasutamine ning pagineerimine.

4.6.1 Vahemälu

Vahemälu rakendatakse liideses andmekihi tasemel, ehk iga SQL'iga tehtud päringu vastus salvestatakse võtme-vastuse paarina vahemällu. Võti sisaldab endas kirjeldust sisu kohta ning päringus sisalduvaid kaasaantud parameetreid. Enne iga SQL päringu tegemist kontrollitakse esmalt võtme olemasolu, ning alles võtme puudumisel tehakse andmebaasi päring. Käesoleva töö kontekstis on vahemälu rakendamine jõudluse parandamise jaoks kasulik, sest kindlatel perioodidel päritakse samasugust informatsiooni tihedamini (kui on nt suurema huviga võistlus või mäng käimas/tulemas).

4.6.2 Pagineerimine

Liidese optimeerimisel on oluline pagineerimine, mis võimaldab suurte andmehulkade ositi laadimist ja tagastamist. Ilma pagineerimiseta tuleks serveril iga päringu korral laadida kogu andmestik, mis põhjustaks väga pikki ooteaegu ning suurt mälukasutust. Pagineerimise abil jaotatakse loendeid tagastavate päringute tulemused "lehekülgedeks"ning kasutajale kuvatakse korraga vaid piiratud arv kirjeid, mille saab arvuliselt kasutaja ise täpsustada päringuparameetrite `pageNumber` ja `pageSize` kaudu. Selline lähenemine aitab oluliselt vähendada nii andmeedastuse mahtu kui ka andmebaasi koormust, võimaldades süsteemil jõudlust säilitada.

4.7 Turvalisus

Laravel pakub erinevaid sisseehitatud mehhanisme, mis aitavad tagada rakenduste turvalisuse nii arenduse kui ka kasutamise seisukohalt. Lisaks järgnevates alapeatükkides loetletud meetoditele on liidese turvalisus tagatud järgmiste omaduste kaudu:

- Päringute koostamisel kasutatakse otsepäringuid koos positsioonilise väärtuste sidumisega. Iga parameeter, mida päring vajab, edastatakse eraldi väärtusena ning seotakse SQL-i kohatäitjatega turvaliselt Laravel Query Builderi abil. Kirjeldatud metoodika maandab SQL süstimise riski ning tagab turvalise andmevahetuse andmebaasiga.
- Vigade korral ei tagasta süsteem kasutajale tundlikku informatsiooni, mis võiks paljastada liidese äriloogikat. Selle asemel kasutatakse struktureeritud veavastuseid ja turvalisi sõnumeid, mis aitavad vältida süsteemi sisemise struktuuri paljastamist.

4.7.1 Märgipõhine autentimine

Rakendusele loodi märgipõhine autentimine, mis võimaldab turvaliselt tuvastada süsteeme, kes kindlaid API lõpp-punkte kasutada soovivad. Märke väljastab Eesti Jalgpalli Liit sidusrühmadele, kellega on organisatsioonil olemas kehtiv koostööleping või ametlik kokkulepe selliseks andmevahetuseks. Autentimine toimib Laraveli sisseehitatud `encrypt()` ja `decrypt()` meetodite kaudu, mis kasutavad AES-256-CBC algoritmi. Krüpteerimine toimub unikaalse võtme alusel, mis on määratud rakenduse konfiguratsioonifailis ja hoitakse serveripoolselt turvaliselt.

Rakenduse andmebaasis loodi spetsiaalne tabel, kuhu talletatakse märgid unikaalse identifikaatori ja krüpteeritud väärtuse paaridena. See võimaldab vajadusel märke kontrollida, kehtetuks muuta või piirata nende kehtivusaega. Süsteemis on implementeeritud kaks eraldi POST tüüpi lõpp-punkti:

- `/createToken` – vastutab märgi loomise eest (nt süsteemisisesele kasutajale või välisele teenusele). Taotlusesse lisatakse juurde kasutaja nimi ning päring saadab kasutajale vastusena krüpteeritud märgi.
- `/useToken` – kontrollib märgi kehtivust. Taotlusesse lisab kasutaja krüpteeritud märgi ning lõpp-punkt saadab vastuses sõnumi, kas märk kehtib või mitte.

Kuigi käesoleva bakalaureusetöö raames loodud liides keskendub vaid lugemispõhistele (GET) lõpp-punktidele, mis ei vaja veel autentimist, on arhitektuuri juba ette valmistatud võimaldamaks turvalist juurdepääsu POST-, PUT- või DELETE-tüüpi operatsioonidele, mida organisatsioon kavatseb tulevikus liidesele juurde lisada. Sellega seoses on oluline, et iga muudatus või sisend valideeritakse ning ligipääsuks nõutakse kehtivat märki.

4.7.2 Sidusrühmade ligipääs rakendusele

Eelmise alapeatüki lõpus oli mainitud, et käesoleva töö raames loodud lõpp-punktide kasutamiseks ei ole kliendi autentimine veel kohustuslik. Klient saab kõiki API GET päringuid teha ilma eelneva tuvastuseta. Süsteem valideerib tagarakenduses sisendi ning tagastab kliendile vastuse JSON-formaadis. Tuleviku tarbeks on arhitektuur ette valmistatud autentimise ja ligipääsukontrolli rakendamiseks - vajaduse korral saab lõpp-punktidele hõlpsalt lisada turvamärgil põhineva autentimise ning võtta märgid klientide seas kasutusele.

4.7.3 Veakäsitlus

Iga sisend, mis jõuab liidese lõpp-punktidesse, läbib serveripoolse valideerimise enne käsitlemist. Selleks kasutatakse Laraveli sisseehitatud validaatorkomponendi reegleid (nt `required`, `string`, `integer`, `in`), et vältida sobimatu või pahatahtlike andmete sisestamist. Parameetrite puhul kontrollitakse vahel ka väärtust (nt kuupäeva puhul selle valiidsust). Juhul, kui kliendi sisend on puudulik või kehtetu või muu viga toimub serveri poolel, siis rakendus tagastab informatiivseid veateateid JSON vormingus, mis sisaldab HTTP olekukoodi ning informatiivset sõnumit.

4.8 Testimine

Rakendusesse loodi lokaalsed integratsioonitestid veendumaks, et API päringud jõuavad õigetesse teenusesse ja süsteemi komponendid suhtlevad omavahel ootuspäraselt. Testides järgitakse ka, et liides tagastab teeseldud andmeid kasutades ootuspärased vastused.

Testimiseks on vaja kasutajal seadistada üles lokaalne MySQL testandmebaas. Projektis on olemas andmebaasi migratsioonid, mida käivitades luuakse samanimelised tabelid, nagu on kasutusel ka *live* keskkonnas. Integratsioonitestide sees lisatakse teeseldud tabelitesse kirjeid ning kutsutakse välja reaalseid liidese lõpp-punkte. Testide tulemustes kontrollitakse õige HTTP koodi tagastamist ning vastuse sisu sobitumist oodatud tulemusega. Testides kasutatakse Laraveli `RefreshDatabase` teeki, mis rakendab andmebaasi migratsioone enne ja pärast iga tehtud testi.

4.9 Dokumentatsioon

Rakendusliidese dokumentatsioon genereeriti automaatselt, kasutades OpenAPI Swagger raamistikku. Swaggeri dokumentatsiooni automaatset genereerimist võimaldavad rakendusliidese kontrollerites defineeritud meetodid. Dokumentatsiooni liideses on võimalik mugavalt testida kõiki rakenduses saadaval olevaid lõpp-punkte. Dokumentatsioon sisaldab endas grupeerituna kõiki saadaval olevaid lõpp-punkte ja nende lühikirjeldusi, päringute ja vastuste näiteid ning parameetrite kirjeldusi (vt Joonis 5 ja Joonis 6). Dokumentatsioon on kättesaadav järgneval veebiaadressil: `https://opendata.jalgpall.ee/api/documentation`

5 Tulemused

Selles peatükis esitatakse töö tulemusi ehk kirjeldatakse valminud funktsionaalsusi, esitatakse jõudlustestimise tulemused ning ettevõttelt saadud tagasiside, kirjeldatakse rakendusega tehtavaid järgmisi samme ning edasiarenduse võimalusi.

5.1 Valminud rakendus ja funktsionaalsused

Käesoleva bakalaureusetöö arendustöö tulemusel valmis uus rakendus, mis sisaldab endas RESTil põhinevat rakendustarkvara liidest. Liides pakub struktureeritud vastustega lõpp-punkte, mis võimaldavad klientidel tarbida Eesti Jalgpalli Liidu andmeid. Rakendus võimaldab kliendil küsida informatsiooni nii kohalike kui ka rahvusvaheliste mängude, meeskondade ning nende koosseisude, mängijate, klubide, võistluste ja statistikate kohta. Lõpp-punktidele saab lisada erisuguseid filtreid, et otsingut täpsustada. Iga olemi kohta on olemas ka üldine lõpp-punkt, kus klient saab otsida nime kaudu olemiga seotud unikaalset identifikaatorit. Seda identifikaatorit võib olla vaja muudes lõpp-punktides parameetrina juurde lisada, niisiis lihtsustab üldine lõpp-punkt vaste leidmist. Üldise informatsiooni saamiseks ning lõpp-punktide lisavõimaluste uurimiseks saab klient uurida Swaggeriga tehtud dokumentatsiooni, kus on ära kirjeldatud iga lõpp-punkti eesmärk ning parameetrid, mida peab ja/või saab päringule juurde lisada.

Rakendus järgib kihilise arhitektuuri põhimõtteid ning on üles ehitatud Laraveli parimaid tavasid ja RESTi standardeid silmas pidades. Rakendus sisaldab endas ka vahemällu salvestamise mehhanismi, mis suurendab jõudlust sagedaste ja/või sarnaste päringute korral. Liides kasutab turvalisuse tagamiseks Laraveli OAuth2 autentimist.

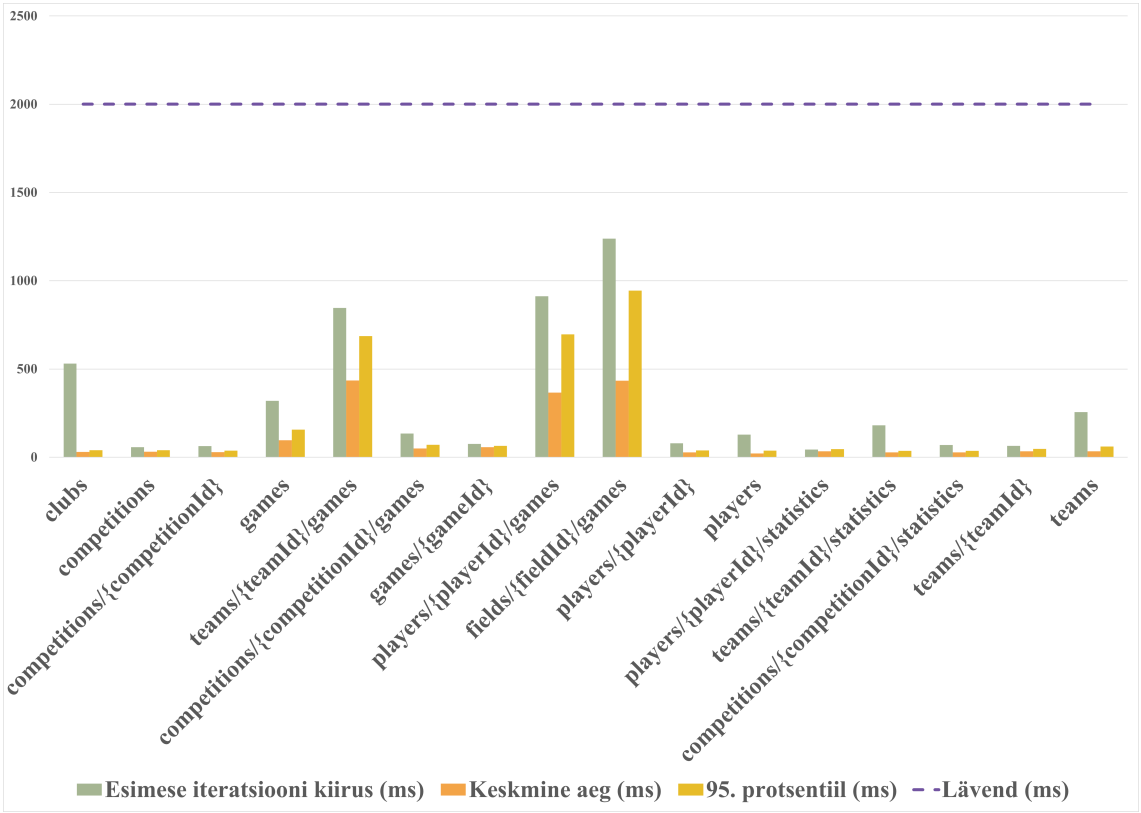
5.2 Jõudlustestimise tulemused

Valminud liidese lõpp-punktide jõudlust mõõdeti vastavalt mittefunktsionaalsele nõudele, mille kohaselt peab süsteem tagastama vastuseid võimalikult kiirelt, kuid mitte rohkem kui

kahe sekundi jooksul. Jõudlustestimise eesmärgiks oli hinnata, kas süsteemi jõudlus vastab püstitatud kriteeriumile ning millised lõpp-punktid vajavad veel optimeerimist. Valideerimiseks mõõdeti `k6` tööriista abil iga lõpp-punkti testimise esimese iteratsiooni vastamisaeg (ehk vastamine vahemälu kasutamata), kõikide päringute keskmine vastamisaeg ning 95. protsentiili vastuse aeg millisekundites (vahemälu kasutades). Testides määrati virtuaalsete simuleeritud paralleelsete kasutajate arv 10 peale, kes samaaegselt liidest koormavad täpselt ühesuguste päringutega. Testide kestus päringu kohta oli 30 sekundit, mille jooksul mõõdetakse süsteemi reageerimisaega ja stabiilsust (nt mitu protsenti päringutest olid edukad). Esitatud tulemustes kajastatakse ainult selliseid mõõtmisi, mille puhul 100% päringutest olid edukad ning iga päring tagastas sisulise vastuse (vastus ei olnud tühi ning loenditega vastuste puhul oli vastuse sees mitu kirjet), et jõudlustestid oleksid võimalikult tõepärased.

Joonisel 4 on esitatud iga lõpp-punkti vastuseajad. Graafikult eristuvad selgelt rohelist värvi tulbad ehk 1. iteratsiooni vastamisajad, mis on võrreldes keskmise- ning 95. protsentiili aegadega suuremad. Tulemus näitab, et vahemälu tabavad vastused tagastatakse märgatavalt kiiremini. Graafikult paistab ka, et pea kõikide lõpp-punktide vastamise ajad jäävad lävendist kahekordselt allapoole; erandiks on lõpp-punkt `fields/fieldId/games`, mille esimene iteratsioon võttis halvimal juhul 1,2 sekundit. Graafikult eristuvad selgemini need lõpp-punktid, mis tagastavad loenditena mängu olemeid ning mille puhul kõik vastamisajad on teiste lõpp-punktidega võrreldes suuremad. Selline tulemus on ootuspärane, sest lõpp-punktide SQL päringud tagastavad rohkelt andmeid ning sisaldavad mitmeid `JOIN` lauseid (mõne päringu puhul üle 10). Kuigi selline vastamisaeg ei ole kriitiliselt kõrge, siis saab tulevikus eesmärgiks seada põhjalikuma analüüsi tegemise, et mõista, kuidas saaks päringuid veelgi rohkem optimeerida.

Arvestades asjaolu, et pea kõik mõõdetud näitajad jäävad alla kahe ning isegi alla ühe sekundi, siis saab jõudlustestide põhjal väita, et süsteem täidab püstitatud mittefunktsionaalse nõude ning on oodatust paremate tulemustega. Järeldada on võimalik ka seda, et vahemällu salvestamise võimekuse lisamine rakendusele aitas kaasa päringute vastuste aja kiirendamisele ning selle vajalikkus on põhjendatud.



Joonis 4. *Jõudlustestimise tulemused*

5.3 Tagasiside ja valideerimine

Töö käigus toimus regulaarne tagasisidestamine nii koolipoolse kui ka ettevõttepoolse juhendajaga. Iga kahe nädala tagant esitles autor tehtud tööd, sai sellele tagasisidet ning omavahel arutati kõiki vahepeal tekkinud küsimusi. Lõplikult valminud toodet testiti ettevõtte IT-osakonna töötajate poolt. Testimisel kontrolliti, et kõik esialgu kokku lepitud funktsionaalsed ning mittefunktsionaalsed nõuded oleksid täidetud (koodis kasutatakse kihilist arhitektuuri, lõpp-punktid toimivad ootuspäraselt ehk tagastavad vajaliku informatsiooni aktsepteeritava aja jooksul jne). Testimisel avastatud vigade, puudujääkide või muude avastuste puhul, mida esialgu ei kaardistatud, saadeti töö paranduste tegemiseks autorile tagasi.

Eesti Jalgpalli Liit testis lõpptoodet kokku kolmel korral. Organisatsiooni tagasisidestamise tulemusel viidi ellu muudatusi ja täiendusi kahel erineval korral.

5.3.1 Tagasiside põhjal tehtud muudatused

Esimesel testimisel peale täiendavaid arutlusi viis töö autor sisse järgnevad muudatused:

- ühte võistluse olemit tagastavat lõpp-punkti muudeti nii, et kui mõnel hooajal ei ole ühtegi võistlusega seotud mängu kirjas, siis tühja nimekirja asemel tagastatakse sisuline tekst;
- ühte võistluse olemit tagastava lõpp-punkti vastuses kajastub, kas tegemist on karikasarjaga või liigaga;
- ühte võistluse olemit tagastava lõpp-punkti vastuses kajastub võistluse tüüp (muru-, saali-, rannajalgpall, *futsal*) ning selle järgi on võimalik kõiki võistlusi filtreerida;
- kõiki võistlusi tagastavas lõpp-punktis on võimalik filtreerida aktiivsuse järgi;
- mängu olemeid tagastavates lõpp-punktides lisati vastuses meeskondade koosseisu kuuluvatele mängijatele tõeväärtus, mis näitab, kes on mängus antud meeskonna kapten;
- muudeti ümber mängija olemi andmete pärimine andmebaasist, sest seda tehti esialgu tabeli kaudu, kus on info puudulikum;
- muudeti statistika olemit tagastavate lõpp-punktide vastuses karikasarja statistikat näitava muutuja nimetus arusaadavamaks;
- täiendati statistika olemit – lisati juurde erinevad keskmised arvud (nt. keskmised väravad mängu kohta, keskmine vaatajate arv) ning võistluste ja meeskondade puhul parimad väravalööjad;
- ühte võistkonna olemit tagastava lõpp-punkti vastuses kajastub ka klubiline kuuluvus;
- kõiki võistkondi tagastavas lõpp-punktis on võimalik filtreerida klubilise kuuluvuse ning aktiivsuse järgi;
- kuna võistkonna juurde lisati klubiline kuuluvus, loodi ka kaks uut lõpp-punkti alamgruppi GET Clubs – üldine lõpp-punkt, kus saab klubisid otsida nime järgi, ning lõpp-punkt, mis tagastab klubi unikaalse identifikaatori kaudu informatsiooni klubi kohta.

Peale loetletud muudatuste implementeerimist valideeriti taas valminud toodet. Eesti Jalgpalli Liidu IT-osakond jäi rahule kõikide tehtud muudatustega. Teisel korral paluti töö autoril veel juurde lisada järgnevalt loetletud kaks erinevat funktsionaalsust:

- esialgu ei kaardistatud ettevõttega, et igal võistlusel on olemas erinev ajaperiood, enne mida ei tohi mängu meeskondade koosseise avalikustada. Töö autor viis sisse mängu olemeid tagastavates lõpp-punktides muudatuse, mis kontrollib, kas päringu tegemise hetkel on võimalik meeskondade koosseise vastuses kuvada;
- esialgu ei kaardistatud ettevõttega, et mängude, võistluste ja statistikate puhul, mille vanusegrupid on alla kaheteistkümne aasta (U12-U8), ei näidata avalikkkusele mängijatega seotud andmeid (sealhulgas võistlustabeleid, väravaid, väravasööte, penalteid, kollaseid ja punaseid kaarte). Töö autor implementeeris seda arvestades muudatused lõpp-punktides, mis tagastavad mängude, võistluste ja statistikate olemeid. Lõpp-punktide vastuste koostamisel kontrollitakse, mis vanusegrupi mängu, võistluse või statistikaga on tegemist, ning kui vanusegrupp jääb U12-U8 vahemikku, kuvatakse mängijatega seotud andmete asemel sisuline tekst.

Peale loetletud muudatuste implementeerimist saadeti liides tagasi organisatsioonile testimiseks. Kolmandal testimisel organisatsiooni IT-osakonna esindaja kinnitas, et loodud lahendus rahuldab kõiki funktsionaalseid ja mittefunktsionaalseid nõudeid ning lõpp-punktide dokumentatsioon teeb lahenduse hõlpsasti integreeritavaks kolmandate osapoolte poolt.

5.4 Rakenduse edasine roll organisatsioonis

Peale lõplikku testimist väljastati lahendus organisatsiooni *live* keskkonda ning alustati Eesti Jalgpalli Liidu andmeid tarbivate sidusrühmade ümber migreerimisega uue rakendusliidese peale. Käesoleva töö kirjutamise ajal on migreerumisprotsess veel kestev, kuid lõppeesmärgiks on viia kõik organisatsiooni andmeid tarbivad sidusrühmad täies ulatuses üle uuele liidesele, et tagada ühtlane andmevahetus kõigi seotud süsteemidega.

Organisatsiooni vahetud edasised plaanid valminud rakendusega on:

- jätkata ning lõplikult migreerida kõik sidusrühmad ümber uue liidese kasutamisele;
- ühildada Eesti Jalgpalli Liidu kodulehe ja uue liidese repositooriumid, et lihtsustada projektihaldust ja vältida dubleerimist äriloogika jagamise kaudu;
- täiustada loodud rakendust ning lisada sinna ka organisatsiooni tundlikumad POST lõpp-punktid, milles saab rakendada lõputöö raames loodud märgipõhist autentimist;

- jälgida edasi lõpp-punktide jõudlust, et tuvastada võimalikud optimeerimise vajadused.

5.5 Edasised arendusvõimalused

Valminud rakendusliides täidab kõik kaardistatud nõuded, kuid liidesele on tulevikus võimalik lisada juurde palju muid funktsionaalsusi ning uusi lõpp-punkte, näiteks:

- valminud rakendusele on planeeritud ka POST-tüüpi lõpp-punktide juurde lisamine, mis bakalaureusetöö raames implementeeritud autentimise ning märkide olemasolu vajavad;
- liidesele saab implementeerida lõpp-punkti, mis võtaks parameetritena kaks erinevat jalgpallimeeskonda ning tagastaks vastuses (kindlal perioodil) omavaheliste mängude võitude, kaotuste ja viikide osakaalud;
- liidese lõpp-punktide vastustele saab lisada juurde väljasid, mis on olemas Eesti Jalgpalli Liidu andmebaasis hetkel, kuid veel ei lisatud või andmeid, mida organisatsioon tulevikus kaardistama hakkab ning soovib klientidele edastada;
- liidesele saab lisada juurde veel erinevat tüüpi statistikaid kajastavaid lõpp-punkte (nt. kohtunike kohta käivaid statistikaid), ning olemasolevatesse lõpp-punktidesse lisada muid arvulisi näitajaid, kui need andmed tehtakse saadavaks – palli valdamisprotsent meeskondade lõikes, löögid värava suunas, löögid raamidesse ja nende osakaal, tõrjed, löögi tabavusprotsent, löögid kastist ja väljaspoolt kasti, nurgalöögid, suluseisud, vaheltvõtud, söödu täpsuse protsent kaitsval ja ründaval väljakupoolel meeskonna lõikes, tsenderdused ja nende täpsus (ehk kui paljud tsenderdused protsentuaalselt jõudsid oma meeskonna mängijani), tehtud vead nii mängijate kui meeskondade lõikes;
- liidesele saab implementeerida lõpp-punktid, mis näitaks mängijate ja meeskondade soorituse numbrilist hinnangut kümne palli süsteemis. Sooritus võib olla ühe mängu või hooaja mängude põhine ning seda võib arvutada palli valdamise, söötude, väravate, kaartide, väravasöötude, penaltite põhjal;
- eelnevalt kirjeldatud lõpp-punkti andmete põhjal saaks luua ennustussüsteemi, mis soorituse põhjal näitaks mõne tulevase mängu võitmise, mängija väravalöömise jms. tõenäosust;

- liidesele saab lisada teenusepõhise logihalduse, mis aitaks jälgida hoiatuste ja vigade tekkimist, kaardistada kasutajate tegevusi ning luua statistikaid kasutajate tehtud päringute põhjal.

6 Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli Eesti Jalgpalli Liidule arendada uus tarkvaraliides, mis oleks valmis avalikuks kasutamiseks kõikide organisatsiooni sidusrühmade poolt neile tarvilikku informatsiooni pärimiseks. Tulemusena valmis liides, mis on laetud üles *live* keskkonda ning Eesti Jalgpalli Liidu sidusrühmad on töö valmimise ajal migreerumas ümber uue liidese peale.

Arendatud süsteem vastab kõigile esitatud funktsionaalsetele nõuetele: liides toetab lugemispõhiseid päringuid, tagastab vastuseid ühtlases JSON struktuuris ning kasutab andmete vahemällu salvestamist. Lisaks on süsteem varustatud märgipõhise autentimise, sisendandmete valideerimise ning informatiivsete veateadetega. Loenditega vastuste puhul rakendatakse pagineerimist. Nõuded mittefunktsionaalsete omaduste osas on samuti täidetud: liidese arhitektuur on modulaarne ja vastab Laraveli parimatele arenduspraktikatele ning töötab turvalise HTTPS-ühenduse kaudu. Liidesele loodi ulatuslik Swaggeri-põhine dokumentatsioon ning lokaalsed integratsioonitestid. Jõudlustestide tulemused näitavad, et liides on võimeline tagastama vastuseid optimaalse ajaga.

Tulemuste põhjal saab väita, et käesoleva töö käigus seatud eesmärk – arendada Eesti Jalgpalli Liidule kaasaegne ja hästi dokumenteeritud tarkvaraliides – on edukalt saavutatud.

Kasutatud kirjandus

[1] Eesti Jalgpalli Liit. [10.05.2025]. URL: https://jalgpall.ee/.

[2] Soccernet. [10.05.2025]. URL: https://soccernet.ee/.

[3] DreamFactory. *Direct Database Access vs. REST APIs: Compare Application Activity*. [10.05.2025]. URL: https://blog.dreamfactory.com/direct-database-access-vs-rest-apis-pros-and-cons-for-application-connectivity.

[4] DreamFactory. *Cybersecurity Risks of Direct Database Connectors*. [10.05.2025]. URL: https://blog.dreamfactory.com/direct-database-connectors-risks.

[5] F. Ben. *What is a Good API Response Time?* [10.05.2025]. URL: https://odown.com/blog/what-is-a-good-api-response-time.

[6] Kanani Nirav. *Top 6 Most Popular API Architecture Styles You Need to Know (with Pros, Cons, and Use Cases)*. [10.05.2025]. URL: https://dev.to/kanani_nirav/top-6-most-popular-api-architecture-styles-you-need-to-know-with-pros-cons-and-use-cases-564j.

[7] *REST API*. [10.05.2025]. URL: https://restfulapi.net/.

[8] *GraphQL*. [10.05.2025]. URL: https://graphql.org/.

[9] Abdul Hakeem. *Best Programming Languages to Develop REST API*. [10.05.2025]. URL: https://blog.tooljet.ai/best-programming-languages-for-rest-api-development.

[10] A. Kataranjee. *The Top Programming Languages for Building APIs*. [10.05.2025]. URL: https://ajee10x.medium.com/the-top-programming-languages-for-building-apis-9da5d1becf43.

[11] W3Schools. *PHP Introduction*. [10.05.2025]. URL: https://www.w3schools.com/php/php_intro.asp.

[12] freeCodeCamp. *What is PHP? The PHP Programming Language Meaning Explained*. [10.05.2025]. URL: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/.

[13] TIOBE. *TIOBE Index*. [01.04.2025]. URL: https://www.tiobe.com/tiobe-index/.

[14] *Laravel Docs*. [10.05.2025]. URL: https://laravel.com/docs/12.x.

[15] REST. *Difference between JSON and XML*. [10.05.2025]. URL: https://restfulapi.net/json-vs-xml/.

[16] IBM. *What is Redis?* [10.05.2025]. URL: https://www.ibm.com/think/topics/redis.

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

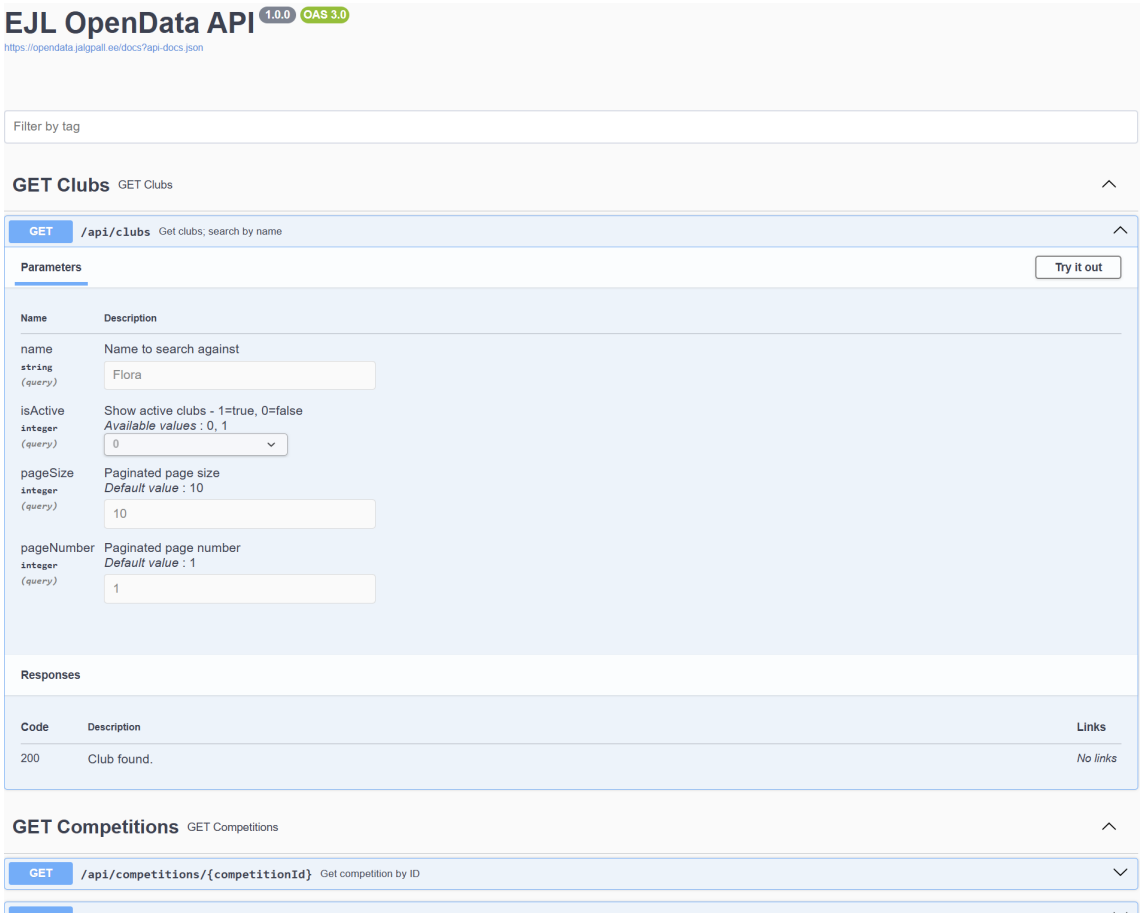
Mina, Emili Kriisa

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Avaliku rakendusliidese loomine Eesti Jalgpalli Liidule", mille juhendajad on Siim Rebane ja Argo Linnaste
    1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
    1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

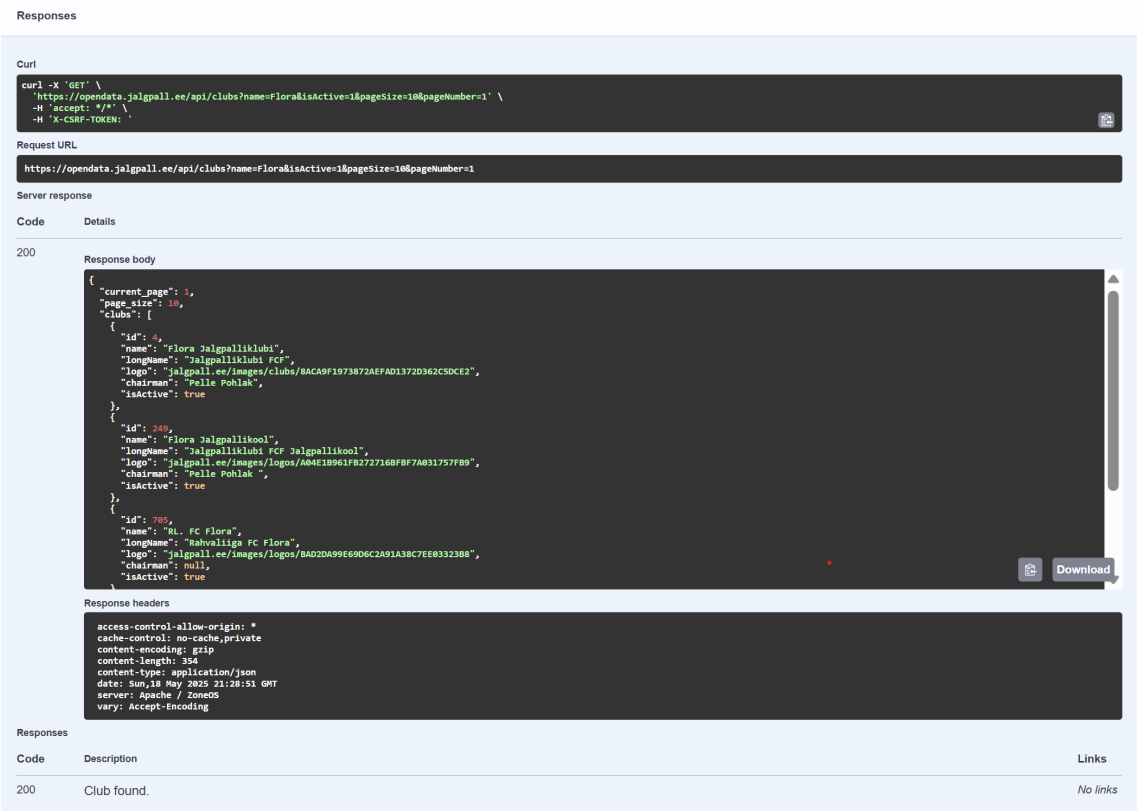
04.06.2025

[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 – Swagger dokumentatsioon

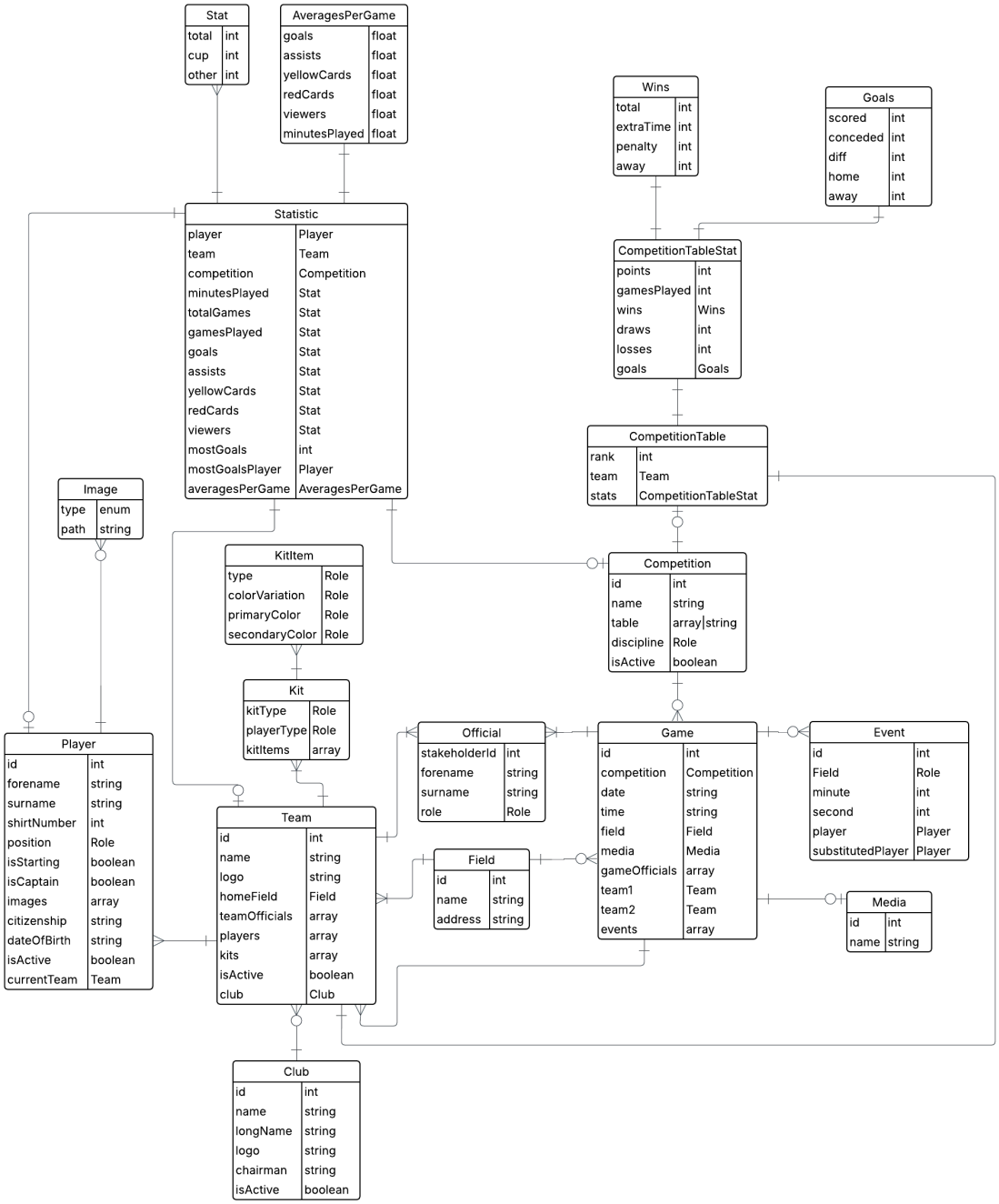


Joonis 5. *Swagger dokumentatsiooni vaade*



Joonis 6. *Vastuse kuvamine Swaggeris*

Lisa 3 – Olemi suhtediagramm andmeväljadega



Joonis 7. *Olemi-suhte diagramm koos andmeväljadega*