TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andri Luik 185208IADB

Rakendus malemõistatuste harjutamiseks ja võistluste korraldamiseks

Bakalaureusetöö

Juhendaja: Märt Kalmo
MSc

Tallinn 2021

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: Andri Luik

17.05.2021

Annotatsioon

Male taktikate arendamine läbi mõistatuste on oluline viis nägemaks taktikaid tõelises mängus. Käesoleva töö eesmärk on luua veebirakendus malemõistatuste lahendamise harjutamiseks ja võistluste korraldamiseks. Hetkel olemasolevad lahendused ei sisalda soovitud funktsionaalsust ja võivad olla tasulised.

Analüüsi käigus uuritakse juba eksisteerivaid maleteemalisi lahendusi, mille käigus täienduvad esialgse visiooni järgi seatud funktsionaalsed nõuded loodavale rakendusele. Samuti leitakse optimaalseimad meetodid nõuete täitmiseks ning valitakse välja tehnoloogiad, mis on antud probleemi lahendamiseks sobivaimad.

Arendusprotsessi käigus luuakse serveri- ning kliendipoolne rakendus kasutades selleks analüüsi tulemustest lähtuvalt tehtud otsuseid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 7 peatükki, 13 joonist, 1 tabel.

Abstract

Application for Practicing Chess Puzzles and Organizing Competitions

Developing chess tactics through puzzles is an essential way to notice tactics in real play. This work aims to create a web application for practicing solving chess puzzles and organizing competitions. Currently, available solutions do not include the desired functionality and might be chargeable.

In the analysis, the author examines the existing chess-themed solutions to supplement the functional requirements set according to the initial vision. The optimal methods for meeting them are found, and the best technologies for implementing the solution are selected.

During the development process, a server-side and client-side application, using the decisions made based on the analysis results, is created.

As a result, a new web application will be created that fulfills all requirements set during the analysis. The application will be an environment where users can develop tactical thinking and find out the most skilled ones among them.

The thesis is in Estonian and contains 29 pages of text, 7 chapters, 13 figures, 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* ehk rakendusliides
CLI	*Command Line Interface* ehk käsurida on tekstipõhine kasutajaliides käskude käivitamiseks
CSRF	*Cross-Site Request Forgery*, saitidevaheliste taotluste võltsimise rünnak
CSS	*Cascading Style Sheets*, keel veebilehtede kujundamiseks ja esitamiseks
CSV	*Comma-Separated Values* ehk komaeraldud väärtustega fail, mis kasutab väärtuste eraldamiseks koma
FEN	*Forsyth-Edwards Notation*, notatsioon malemängu lauapositsiooni kirjeldamiseks
HTML	*Hypertext Markup Language* ehk hüperteksti märgistuskeel, keel veebilehtede märgendamiseks
HTTP	*Hypertext Text Protocol* ehk hüperteksti edastusprotokoll, protokoll teabe edastamiseks arvutivõrkudes
JSON	*JavaScript Object Notation*, andmevahetusvorming
MVC	*Model-View-Controller* ehk mudel-vaade-kontroller, tarkvara arhitektuurimuster
NPM	*Node Package Manager*, JavaScripti teekide haldusvahend
RDMS	*Relational Database Management System* ehk andmebaasi haldussüsteem
REST	*Representational State Transfer*, tarkvaraarhitektuuri stiil, mis seab veebirakenduse loomisele kindlad reeglid
URI	*Uniform Resource Identifier* ehk ühtne ressursiidentifikaator, sõne infoallika üheseks määramiseks veebis

Sisukord

Jooniste loetelu

Tabelite loetelu

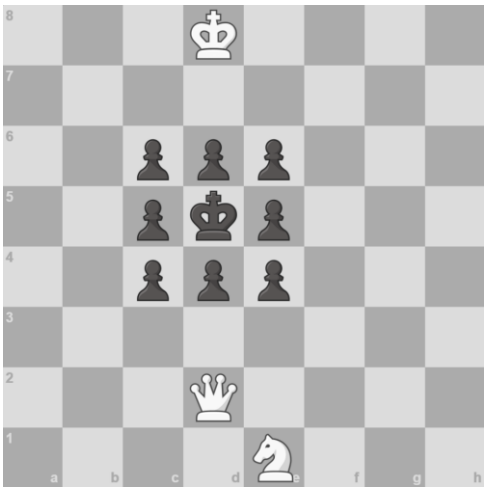
1 Sissejuhatus

1.1 Taust ja probleem

Maleprobleemi all mõistetakse malepositsiooni, mis testib lahendaja suutlikkust leida korrektne lahendus. See tähendab oskust näha ja läbi viia jada käike, mis võimaldavad teenida ajalist eduseisu, takistada vastase plaane, võita tema mängunuppe, või ta alistada. Rohkete taktikatega mängud pakuvad samasugust rahuldust nagu teiste sportide suurejoonelised hetked, näiteks kodujooks pesapallis või pealtpanek korvpallis. Selliseid mänge peetakse ilusateks ja imetlusväärseteks. Neid korratakse üle maailma uuesti ja uuesti ning kasutatakse õppematerjalidena [1], [2], [3].

Mängija malemängu kvaliteedi määrab tema mõttekäigu kvaliteet ehk võime kaaluda potentsiaalse järgmise käigu tugevaid ja nõrku külgi. Algajatele maletajatele võivad tunduda kogenud mängijate poolt võitmiseks kasutatavad käikude jadad võimatult keerulised. Siiski enamasti koosnevad need levinud ideedest ja printsiipidest (taktikatest ja strateegiadest), mis on leidlikul viisil omavahel ühendatud. Taktikate tähtsust tuleks mõista ja õpetada kõikidele malehuvilistele. See teeb neist paremad maletajad [3], [4].

Maleprobleeme on kokku kaks liiki: malekompositioonid ja malemõistatused. Malekompositsioonid (ing. k. *chess composition*) on koostatud inimese või arvuti poolt. Nad sisaldavad stipulatsiooni, mis määrab positsiooni oodatava tulemuse, näiteks matt kahe käiguga. Tulemuseks on alati mängu lõpetav käik ehk matt või viik. Kompositsioonides esinevad tihti positsioonid, milledeni jõudmine tõelises mängus on ebarealistlik (Joonis 1). Ametlikke malekompositsioonide lahendamise võistlusi (ing. k. *chess solving competitions*) peetakse iga aasta päris palju ja ka rohkete osavõtjatega [5]. Näiteks rahvusvahelisel lahendusvõistlusel (ing. k. *International Solving Contest*) osales 2019. aastal 648 lahendajat 49 riigist [6]. Eesti nende hulgas aga mitte, võttes viimati osa 2010. aastal.



Joonis 1. Malekompositsioon: matt nelja käiguga.

Malemõistatused (ing. k. *chess puzzle*) on kompositsioonidega võrreldes tüüpilisemad ja enamasti lihtsamad. Malemõistatused esitlevad positsioone, mis on võetud tõeliselt mängitud mängudest või vähemalt tunduvad realistlikud (Joonis 2). Nende tulemuseks on tavaliselt samuti mängu lõpetav käik või materiaalne üleolek vastasest [7], [8], [9].

Joonis 2. Malemõistatus: valge võidab materjali.

Malemõistatuste lahendamise võistlusi peetakse aga vähe. Seda sel põhjusel, et malemõistatused on kompositsioonidest märksa lihtsamad ning enamasti nende lahendamist ainult harjutatakse. 2021. aasta alguses toimusid teist aastat veebisaidi Chess.com algatatud maailmameistrivõistlused malemõistatuste lahendamises (ing. k. *Puzzle Battle World Championship*) [10]. Võistlus toimub nende veebikeskkonnas paari kaupa võisteldes ning osalema võib kandideerida igaüks, kuid kõrge lävendi tõttu jõuavad lõppvooru ainult väga tugevad mängijad. Puhtalt huvilistele suunatud sarnaseid sündmusi on raske leida.

Malekompositioonide eesmärk on panna lahendajat tõsiselt proovile. Nende lahendamine võtab enamasti kauem aega kui malemõistatuste puhul. Seevastu malemõistatuste eesmärgiks on positsioonides peituvate taktikate nägemine ja tundma õppimine, et osata märgata neid ka tõelises mängus. Sel eesmärgil käsitletakse käesolevas töös ainult malemõistatusi.

Male on üks vanimaid lauamänge maailmas [11]. Kuigi läbi sajandite on ta mõneti muutunud, mängitakse teda tänapäeval endiselt väga palju. Male populaarsus hakkas maailmas eriti kasvama 2020. aastal alanud pandeemia ajal. Sama aasta oktoobris liitusid mänguga miljonid uued mängijad, kes said inspiratsiooni uuest maleteemaliset telesarjast „Lipugambiit“ (ing. k. *The Queen's Gambit)*, kui avaldati selle esimese osa [12].

Ka Eestis saab malet lugeda populaarseks. Eesti Maleliidu alla kuulub 54 siin tegutsevat maleorganisatsiooni [13]. Maleliidu kodulehel sündmuste kalendris on märgitud võistlusi pea igale nädalale üks või rohkemgi [14].

Võistlemine on põnev ning konkurentsi kogemine õhutab tegema veelgi paremini ja tõstab teema vastu huvi. Males võistlemiseks on võimalusi kuhjaga, kuid selleks võimalusi malemõistatuste lahendamises vaid käputäis. Niivõrd olulise oskuse arendamise jaoks puudub keskkond, kus oleks olemas soovitud funktsioonid ning huvilised saaksid koguneda.

1.2 Eesmärk

Käesoleva töö eesmärk on analüüsida erinevaid võimalusi antud probleemi lahendamiseks ning valida sobivaim. Praktilise osa eesmärk on jõuda lõpprakenduseni, mis toimib seatud kriteeriumite järgi.

1.3 Metoodika

Töö algab esialgsete nõuete paika panemisega probleemile otsitavale lahendusele. Need nõuded kirjeldavad, mida minimaalselt lahendusest oodatakse, ning neid arvesse võttes uuritakse, kas eksisteerib reaalselt juba nõuetele vastavaid rakendusi. Uurimise käigus tuuakse välja uurimisobjektide tugevad ja nõrgad küljed ning selgitatakse valikut luua uus lahendus. Uurimistulemuste põhjal täiendatakse esialgselt seatud nõudeid ning leitakse sobivad tehnoloogiad lahenduse elluviimiseks.

Lõputöö teises pooles kirjeldatakse rakenduse arenduse protsessi nii kliendi- kui ka serveripoolelt. Viimasena võetakse kokku tehtud töö tulemused.

2 Probleemi analüüs

2.1 Lahendusele esialgselt seatavad nõuded

Käesolevas alapeatükis tuuakse välja nõuded, mis peaksid olema täidetud tõstatatud probleemi minimaalses lahenduses.

2.1.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded panevad paika kriteeriumid kirjeldamaks tegevusi, mida süsteem peab olema võimeline tegema [15]:

- Rakenduse avamisel peaks kasutaja saama valida, kas ta soovib malemõistatusi harjutada või nende lahendamises võistelda.
- Harjutamise režiimis peaks kuvatama malelaud koos malenditega. Järgmise mõistatuse laadimiseks vajutab kasutaja vastavat nuppu.
- Võistlemise režiimis peaks kasutajale kuvatama hetkel käimas ja eesootavad sündmused, mille hulgast saab ta valida endale sobiva.
- Võistluse loomine peaks olema võimalik kõigile, kel on olemas konto. Luua saab privaatseid (parooliga kaitstuid) kui ka avalikke võistlusi, millega võib liituda vähemalt 2 osalejat.

2.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded keskenduvad süsteemi käitumise täpsustamisele, mitte konkreetsete funktsioonide või omaduste kirjeldamisele [15]:

- Ligipääs kõikidele rakenduse funktsioonidele on täielikult tasuta.
- Rakenduses navigeerimine peab olema hästi mõistetav kõigile.

2.2 Olemasolevate lahenduste otsimine ja uurimine

Maleteemalisi rakendusi on loodud mitmeid, nii veebi- kui ka mobiilirakendusi. Male ühendab neid rakendusi, kuid kõik nad ei ole loodud täitma ühist eesmärki. Eesmärke on

erinevaid: uudiste jagamine, male mängimine, mõistatuste lahendamine, inimeste ühendamine, male õppimine jne. Enamik rakendusi täidavad mitmeid eesmärke korraga, enda jaoks sobilikku tuleb lihtsalt osata otsida. Järgnevates lõikudes on selgitatud, kuidas leiti uurimiseks teemakohaseid rakendusi, ning analüüsitakse nende vastavust probleemi oodatavale lahendusele.

Eksisteerivate rakenduste leidmiseks kasutas autor Google otsingumootorit [16] veebirakenduste leidmiseks ning Google Play rakenduste otsingut [17] mobiilirakenduste leidmiseks. Otsingusõnadena või -fraasidena kasutati nii eesti- kui ka ingliskeelseid väljendeid. Täieliku otsingusõnade ja -fraaside loendi leiab töö lisadest (Lisa 2).

Kokku uuriti analüüsi käigus 45 erinevat rakendust (Lisa 3). Tulemustesse kaasati ainult need, mis sisaldavad malemõistatuste lahendamise funktsiooni. Analüüsi käigus otsiti vastuseid järgmistele küsimustele:

- Kui paljudes rakendustes saab piiramatult tasuta harjutada malemõistatusi?
- Kui paljud rakendused sisaldavad malemõistatustes võistlemise funktsiooni?

Tulemustest selgus, et enamik rakendustest võimaldas piiranguteta tasuta harjutamist. Nendele rakendustele ei ole seatud kasutamisajast või liikmestaatusest sõltuvaid piiranguid. Tasulisteks tundusid osutuvat just koolitamise teenust pakkuvad või laia funktsioonide valikuga rakendused, millede arenduse taga on terved palgalised meeskonnad.

Malemõistatuste lahendamises võistlemist võimaldasid analüüsitud rakendustest 6, mis ka kõik võimaldasid tasuta harjutamist. Siiski nendest 4 lubasid pidada ainult duelle. Ülejäänud kaheks rakenduseks on:

1. chess.cool – lihtsa ning arusaadava disainiga lehekülg, mis sisaldab peale malemõistatuste ka muid maleteemalisi funktsioone. Rakendusel on suured puudused: võistlused on kõik avalikud ning kasutajad ise neid luua ei saa. Samuti on korraga käimas alati ainult 1 võistlus, milles kasutatavate mõistatuste teemad võivad varieeruda.
2. chesscup.org – ilusa kasutajaliidesega rakendus, mis pühendub ainult malemõistatuste lahendamisele. Võrreldes chess.cool leheküljega, võimaldab chesscup.org märksa enam. Võimalik on pidada nii duelle kui ka suurema osalejate arvuga võistlusi.

Korraga võib neid käia suur hulk ning igaühel on võimalik alustada uut, mis võib olla nii avalik kui ka privaatne. Siiski puudusena koosneb iga turniir kindla ajalise pikkusega fikseeritud arvust voorudest. Kasutajale ei ole jäetud piisavalt võimalust kohandada turniiri peamisi reegleid oma äranägemise järgi.

Eksisteerivate rakenduste peamiseks puuduseks on malemõistatuste lahendamises võistluste korraldamise funktsiooni puudumine. Need üksikud rakendused, millel see funktsioon on olemas, piiravad osavõtjate arvu kõigest kahele või jätavad vähe võimalusi seadistada võistluse reegleid.

3 Loodava lahenduse planeerimine

3.1 Nõuete täiustamine

Esialgselt seatud nõuded panid paika, mis funktsioonid peavad kindlasti lahenduses olemas olema, et töös käsitletav probleem saaks lahendatud. Siiski ainuüksi nende nõuete täitmisel ei oleks rakendus tõsiseltvõetav. Esialgsetele lisanduvad nõuded, mis täpsustavad või täiustavad juba olemasolevaid, andes loodavale rakendusele lisaväärtust ja aitavad kaasa kasutajate arenemisprotsessile ning parandavad kasutajakogemust.

Rakendusele lisanduvad funktsionaalsed nõuded:

- Kasutajad peavad saama süsteemis luua isiklikku kontot. Kontoga identifitseeritakse ühte isikut, kelle rakenduses salvestatavad toimingud seotakse antud kontoga.
- Peale mõistatuse ebakorrektset lahendamist peaks olema võimalik näha korrektset lahendusviisi. Nii saab kasutaja analüüsida ja õppida, mis oli talle tähelepanuta jäänud ning oskab loodetavasti tulevikus sarnaseid lahendusi ära tunda.
- Mõistatused peavad olema jagatud nende keerulisuse põhjal raskusastmetesse ning mängija saab harjutades valida, millisesse raskusastemsse kuuluvaid mõistatusi talle kuvatakse.
- Võistlusi saavad luua ja nendel osaleda vaid registreerunud kasutajad. Vastasel juhul tuleks kuvada edetabelis osaleja anonüümsena, mis ei ole mõtekas.
- Võistluse loomisel valib korraldaja võistlusele nime, mille järgi teda on võimalik nimekirjast otsida. See teeb lihtsamaks soovitud sündmuse leidmise võimalikust pikast nimekirjast.
- Võistluse korraldaja seab paika selle kestvuse ning saab määrata ka võistleja poolt lahendatavate mõistatuste arvu. Kestvuse lõppemisel või soorituse lõpetamisel kuvatakse võistlejale edetabel.
- Võistlemise ajal ebakorrektse lahenduse pakkumisel lahendust ei näidata. Võistlustules ei jääks lahendajale aega neid uurida ning nii saab ta keskenduda lahendamisele.

Rakendusele lisanduvad mittefunktsionaalsed nõuded:

- Ilma isikliku kontota kasutaja kohta andmeid ei salvestata.

3.2 Malemõistatuste kogumine

Malemõistatuste kogumisel tuleb arvestada rakendusele seatud nõuetega. See tähendab, et kasutuskõlblikud on ainult need mõistatused, millede kohta on piisavalt andmeid, et täita seatud nõudeid. Arvestades käesoleva töö eesmärki ning võttes kokku peatükkides 2.1 ja 3.1 kirjeldatud nõuded, on vajalikud järgmised andmed:

- Malemängu positsiooni esitus mingil kujul, millest loetakse välja info positsiooni kuvamiseks mängijale.
- Lahenduskäik, et kontrollida mängija tulemust ning vajadusel kuvada talle korrektset lahendust.
- Mõistatuse raskusaste, et mängija saaks kohandada kuvatavaid mõistatusi talle sobiva raskusastme järgi.

Antud probleemi jaoks sobiks väga hästi lichess.org avatud andmebaas [18], mis sisaldab üle ühe miljoni kirje koos vajalike andmetega. Andmebaas on CSV (*Comma-Separated Values*) formaadis. Iga kogumis leiduv kirje vastab ühele mõistatusele ning kuvab nõutud andmeid järgmiselt:

- Malemängu positsiooni esitus on kujutatud FEN (*Forsyth-Edwards Notation*) sõnena. FEN sisaldab kogu vajalikku teavet, et taaskäivitada malemängu kindlalt positsioonilt, näiteks nuppude paigutus ning järgmise käigu sooritaja.
- Lahenduskäik on kuvatud samuti sõnena. Iga sõne koosneb tühikutega eraldatud nelja- või viiekohalisetest alamsõnedest. Alamsõnede kaks esimest sümbolit tähistavad maleruutu, millel asuvat nuppu liigutatakse. Kaks viimast sümbolit tähistavad ruutu, kuhu see nupp liigutatakse. Alamsõne lõpus võib esineda üks lisasümbol, mis tähistab malendit, milleks edutatakse ettur, kui see jõuab malelaua viimasele reale.
- Raskusastet kujutatakse numbriliselt. Igal mõistatusel on arvuline reiting ning mida kõrgem see on, seda keerulisem on mõistatus. Andmebaasist on võimalik leida kõrgeima ja madalaima reitinguga mõistatused ning nende reitingute vaheline piirkond jagada raskusastmeteks.

3.3 Abistavate teekide leidmine

Tehtava töö hulga vähendamiseks on mõistlik leida abistavaid teeke, mis sisaldavad soovitud funktsioone. See päästab arendajale aega, sest töö on juba kellegi teise poolt ära tehtud.

Käesoleva töö raames luuakse maleteemaline rakendus, seega on kindlasti vajalik kuvada kliendiaknas kasutajatele malelaud koos malenditega, mis peab meeles malelaual kujutatud positsiooni. Kasutades olemasolevate rakenduste otsimiseks kasutatud otsingufraase NPM (*Node Package Manager*) teegihalduri kodulehe otsingureal [19], leiab soovitud eesmärki täitvaid teeke:

- Cm-chessboard [20];
- Chessground [21].

Mõlemad teegid pakuvad lahendust kuvada malelauda koos malenditega ning on kasutatavad litsentside alusel, mis lubavad pakutvat tarkvara tasuta kasutada, muuta ja jagada. Võrreldes Cm-chessboard teegiga pakub Chessground rohkelt erinevaid lisafunktsioone, mis aga kõik on loodud pidades silmas lichess.org rakenduse vajadusi. Cm-chessboard täidab täpselt käesoleva töö raames loodava rakenduse vajadusi ning selle paigaldamine on lihtsam. Seepärast osutub valituks Cm-chessboard.

Lisaks on vajalik mängija käikude reeglitele vastavust valideerida. Mõistatuse lahendamisel ei võeta arvesse lahenduskäiku, mis sisaldab ebareeglipäraseid käike. Selle jaoks on samuti erinevaid lahendusi:

- chess.ts [22];
- node-chess [23];
- cm-chess [24];

Chess.ts ja Cm-chess on mõlemad teegi Chess.js [25] edasiarendused. Chess.ts on Chess.js ümberkirjutus TypeScripti keelde ning Cm-chess lisab juurde mõned funktsioonid. Node-chess on väga sarnane kahe eelnevalt välja toodud teegiga, kuid see ei sisalda võimalust laadida programmi malepositsiooni FEN kujul, mis aga on lähtuvalt kogutud malemõistatuste kohta olemas olevate andmete tõttu vajalik. Cm-chess ja Chess.ts on omavahel võrdsed teegid ning valik langeb nendest viimase kasuks seetõttu, et staatiliselt tüübitud Typescripti keel omab Javascripti ees olulisi eeliseid nagu vigade

varasem avastamine. Rakenduse loomisel saab tõenäoliselt olema kliendipoolseks programmeerimiskeeleks samuti Typescript.

4 Tehnoloogiate analüüs

Käesoleva töö väljund on veebirakenduse loomine, mis vastab probleemi analüüsis seatud nõuetele. Eduka rakenduse üks eeldusi on kindlasti tööks vajalike vahendite läbimõeldud valik, mis arvestab varasemalt seatud nõudeid ning probleemi ja selle lahendusviisi isepärasusi. Rakenduse arendamiseks sobivate tehnoloogiate kaalumisel on vaadeldud vaid kaasaegseid ning kestva toega tehnoloogiaid, millel on lai kasutajabaas. Valitud tehnoloogiad ei tohi ka olla tasulised. Järgnevalt on välja toodud potentsiaalsed tehnoloogiad rakenduse loomiseks ning nende hulgast tehtud põhjendatud valik.

4.1 Programmeerimiskeele valik

Eelistatavalt peaks valitav keel olema staatiliselt tüübitud keel ehk muutuja tüüp on teada kompileerimise ajal. Selle eeliseks on arendusprotsessis võimalike vigade varajane avastamine, programmeerijale koodi mõistetavamaks tegemine ning mitmed arenduskeskkonnad oskavad nende keelte puhul pakkuda arendustööd kiirendavaid lisafunktsioone, nagu näiteks klassi meetodite soovitamine. Staatiliselt tüübitud keelte kõrval on olemas ka dünaamiliselt tüübitud, mis kontrollivad muutuja tüüpi programmi käivitamisel, või tüüpimata keeled. Need aga mainitud eeliseid ei paku. Sellel põhjusel jäävad valikust välja sellised keeled nagu Python, PHP ja Javascript.

Valikus on veel keeled Typescript, C# ja Java. Typescriptiga võrreldes on C# ja Java palju küpsemad keeled. C# on loodud Microsofti poolt ning Java on Oracle omandis. Sellest valikust sõltub ka tarkvara arenduse raamistiku valik. Mõlema keeled on objekt-orienteeritud, platvormist sõltumatud, tugevalt tüübitud, suure kasutajaskonna ning laia tööriistade valikuga. Mõlemad keeled sobiksid käesoleva töö eesmärgi saavutamiseks suurepäraselt. Seetõttu tehakse valik võttes arvesse rakenduse ainsa arendaja (töö autori) eelistusi. Võrreldes Javaga on tal tunduvalt vähem kogemusi C# keele ja Microsofti arendustarkvaradega ning sel põhjusel valitakse serveripoolseks keeleks Java.

4.2 Tarkvara arenduse raamistiku valik

Valitavale tarkvara arenduse raamistikule erilisi nõudeid seatud ei ole, mis tõstaksid kindla raamistiku kohe esile. Vaja on vastu võtta päringuid, neid töödelda, suhelda andmebaasiga ja lõpuks saata välja vastus. Raamistik võiks samuti toetada REST (*Representational State Transfer*) arhitektuuri rakendamist.

Valikut kitsendab serveripoolseks arenduseks valitud programmeerimiskeel, milleks sai Java. Kõige populaarsem Java arendusraamistik on Spring ning see sisaldab soovitud funktsioone, sobides väga hästi antud töö tegemiseks. Seetõttu langeb valik Springi komponendi Spring MVC (*Model-View-Controller*) peale, mis surub veebirakendusele peale MVC arhitektuuri ning pakub mugavusi ja funktsionaalsust. Lisaks võetakse rakenduse püstipaneku lihtsustamiseks kasutusele Spring Boot tööriist. Spring Boot eeliseks on automaatselt loodud, levinud tavadel põhinev konfiguratsioon. See lühendab oluliselt arendusele kuluvat aega ning suurendab tootlikkust [26], [27].

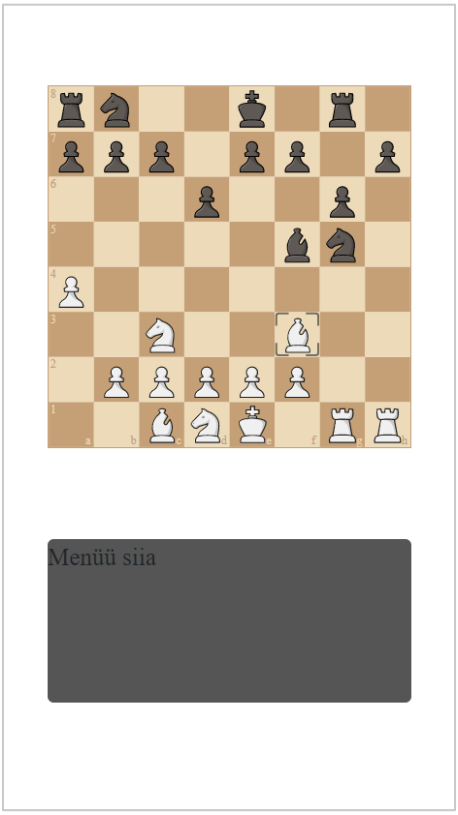
4.3 Kasutajaliidese raamistiku valik

Kliendipoolse arendusraamistiku valikusse kuuluvad tänapäeval enim kasutatud ja hinnatud raamistikud Angular, React ja Vue [28], [29]. Neid kõiki uuendatakse aktiivselt, andes regulaarselt välja uusi versioone ning hooldades olemasolevaid. Iga raamistiku taga on suur kogukond ja see areneb pidevalt. Nad ka jagavad omavahel mitmeid sarnasusi, kuid ühelgi mainitud raamistikul ei esine määravat puudujääki või eelist, mis langetaks otsuse tema kasuks või selle valikust hoopis välja jätaks.

Otsuse langetamiseks on oluline testida vaate poolel kõige tähtsamat komponenti, milleks on malelaud. Peatükis 3.3 valiti selle kuvamiseks välja selleks vastav teek. Teeki prooviti kasutada ja kuvada esmalt Angular komponendina (Lisa 4).

Angulari kohta on hea, et see on juba kirjutatud TypeScriptis, mis tagab tüüpide (primitiivide, liideste jms) toetamisel suurema turvalisuse. See aitab vigu koodi kirjutades või hooldustoiminguid tehes varakult tabada [30]. Samuti paneb raamistik paika reeglid, kuidas struktuuri hoida ja ilusat koodi kirjutada. Rakendus valmib koostades HTML-il (*Hypertext Markup Language*) põhinevaid malle (*templates*), mille haldamiseks on komponendiklassid (*components*). Rakenduse loogika lisatakse teenusklassidesse (*services*) ning kõik need seotakse omavahel moodulitesse (*modules*) [31].

Malelaua testimise tulemus brauseris on kuvatud mobiilivaates, et joonis ei oleks liiga suur (Joonis 3).



Joonis 3. Malelaua kuvamine kliendiaknas kasutades cm-chessboard teeki.

Teegi kasutamine oli lihtne ning see täidab ootusi: mängulaua positsiooni peetakse meeles ja selle konfigureerimiseks on lai valik võimalusi, et rakendus sisaldaks eesmärgipäraseid funktsioone. Ülejäänud raamistike testimisel ei nähtud enam mõtet ning valik tehtigi Angulari kasuks.

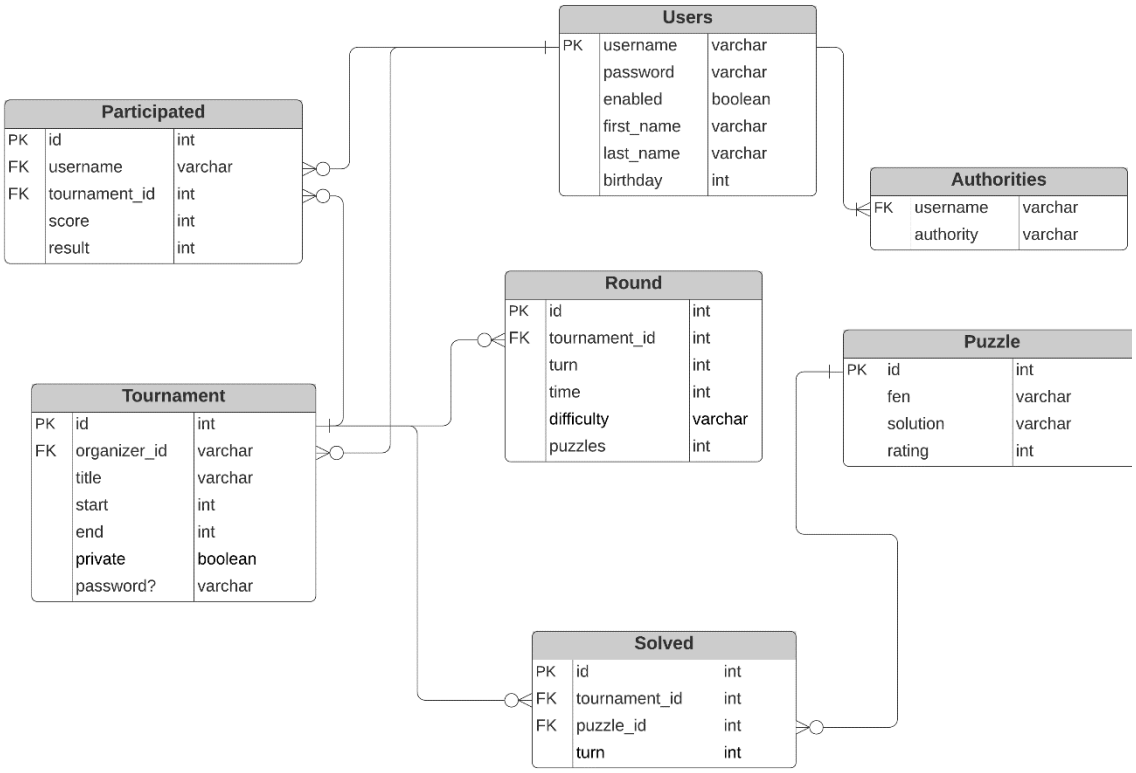
4.4 Andmebaasisüsteemi valik

Andmebaasi valik tehakse relatsiooniliste andmebaasisüsteemide hulgast, sest tänapäeval enim eelistatuimad süsteemid põhinevad relatsioonilisel mudelil [32].

Töö skoopi jäävas andmebaasis on 7 tabelit (Joonis 4):

1. *Users* – rakenduses loodud kontod ning sellega seotud andmed.
2. *Authorities* – kontole antud roll, mis määrab näiteks juurdepääsuõigused. Antud rakenduses on õiguste haldamine väga lihtne, kuna kokku on ainult kaks rolli.
3. *Participated* – andmed võistlustel osalenud konto ja tema tulemuste kohta.

4. *Tournament* – kindla konto poolt loodud malemõistatuste lahendamise võistlus.
5. *Solved* – andmed võistlustel kasutatud malemõistatuste kohta.
6. *Puzzle* – tabel, mis sisaldab kõiki malemõistatusi.
7. *Round* – tabel, mis sisaldab võistlustega seotud raundide andmeid.



Joonis 4. Andmebaasi olemi-suhte diagramm.

Tuntud relatsioonilisi andmebaase on mitmeid: SQLite, MSSQL, Oracle, MySQL, PostgreSQL jne. Käesoleva töö skoopi arvestades ei ole vaja teha keerukaid päringuid ja andmete hulk ei ole samuti suur, mistõttu peaksid kõik loetletud süsteemid probleemideta saama hakkama soovitud rakenduse andmete haldamisega. Valiku tegemiseks otsustati testida süsteemidest esmalt lihtsasti üles seatavat, tasuta RDMS-i (*Relational Database Management System*) SQLite:

- SQLite on „iseseisev“, sest tal on väga vähe sõltuvusi ehk sõltub välistest teguritest vähe. SQLite töötab mis tahes operatsioonisüsteemil ning selle ehitamiseks pole vaja erilisi tööriistu [33].
- SQLite on serverita ehk andmebaasile ligipääsemiseks ei ole olemas vahepealset serverit, läbi mille teised andmebaasisüsteemid andmete kättesaamiseks peavad suhtlema. Andmebaasi ülesseadmine on seetõttu palju kiirem, sest serverit pole vaja paigaldada ega konfigureerida [34].

Need kaks põhjust aga ainuüksi ei anna piisavalt aimu, kas SQLite sobib nõutud andmete hoiustamiseks. Andmemudelist on näha, et andmebaas ei saa sisaldama keerulisi andmeid, mis on kõik kõige tavalisemate andmetüüpide kujul. Seega ei tohiks tekkida probleeme andmete paigutamisega. Näiteks malemõistatuste andmed tulevad välisest failist, mis imporditi CSV formaadis olevast üle miljoni kirjega kogumist. Selleks kirjutati programm, mis täitis oma ülesande kõigest kuue sekundiga. Põhjalikumaks testimiseks täideti ülejäänud tabelid mõnede väljamõeldud andmetega. Kuna rakenduses peetakse võistlusi, võib aeg osutuda oluliseks. Andmebaasis jooksutati mõnda lõpplahenduses tõenäoliselt sagedasti ettetulevat päringut. Eesmärgiks seati, et päringu täitmine ei võta kauem kui 0,1 sekundit:

- Joonisel 5 on päring, mida läheb vaja juhuslikult valitud mõistatuste saamiseks, millede reiting kuulub kindlasse raskusastmesse ning korraga päritakse 20 mõistatust. Päringu täitmine võttis 0,06 sekundit.

```
SELECT p.id, p.fen, p.solution, p.raiting FROM puzzle
WHERE rating BETWEEN 2000 AND 2399
ORDER BY RANDOM() LIMIT 20;
```

Joonis 5. Päring mõistatuste jaoks.

- Joonisel 6 on päring, mida on vaja võistluste menüü vaate loomiseks, kus kasutajad saavad näha hetkel toimuvaid ja järgneva nädala jooksul algavaid võistlusi. Vaates kuvatakse andmeid võistluse kohta, nagu nimi, algus- ja lõpuaeg, seni registreernud osalejate arv, kui palju raunde ta sisaldab ja kas tegemist on parooliga kaitstud sündmusega. Lisaks on vaate jaoks vajalik pärida jaoks infot, kas kasutaja on juba registreerunud. Päringu täitmine võttis 0,048 sekundit.

```
SELECT t.id, t.organizer_username AS 'organizer', t.title, t.start,
       t.end, t.is_private AS 'isPrivate', t.password,
       COUNT(DISTINCT r.id) AS 'rounds',
       COUNT(DISTINCT p.id) AS 'participantsNo',
       (SELECT 1 FROM participated p2 WHERE p2.tournament_id = t.id
       AND p2.participant = :username) AS 'isRegistered'
FROM tournament t
LEFT JOIN participated p ON t.id = p.tournament_id
JOIN round r on t.id = r.tournament_id
WHERE t.start BETWEEN :current AND :after OR t.start <= :current AND
       t.end >= :current
GROUP BY t.id;
```

Joonis 6. Päring kindla võistluse ja nende osalejate kohta.

Joonisel 5 olev päring oli aeglasem kui joonisel 6 välja toodud päring, isegi et ta tundub olevat palju lihtsam. Seda seetõttu, et mõistatuste tabelis on üle miljoni kirje, mida on

palju rohkem kui ülejäänud tabelitesse sisestatud võltsandmed. Esimese päringu puhul kiirendas ka vastuse saamist väljale „rating“ seatud indeks.

Kuna SQLite vastas oodatud tulemustele, siis leiti, et ta sobib hästi rakenduse andmebaasisüsteemiks. Otsustati, et teisi süsteeme üle testima ei hakata ning jäädakse SQLite juurde.

5 Teostus

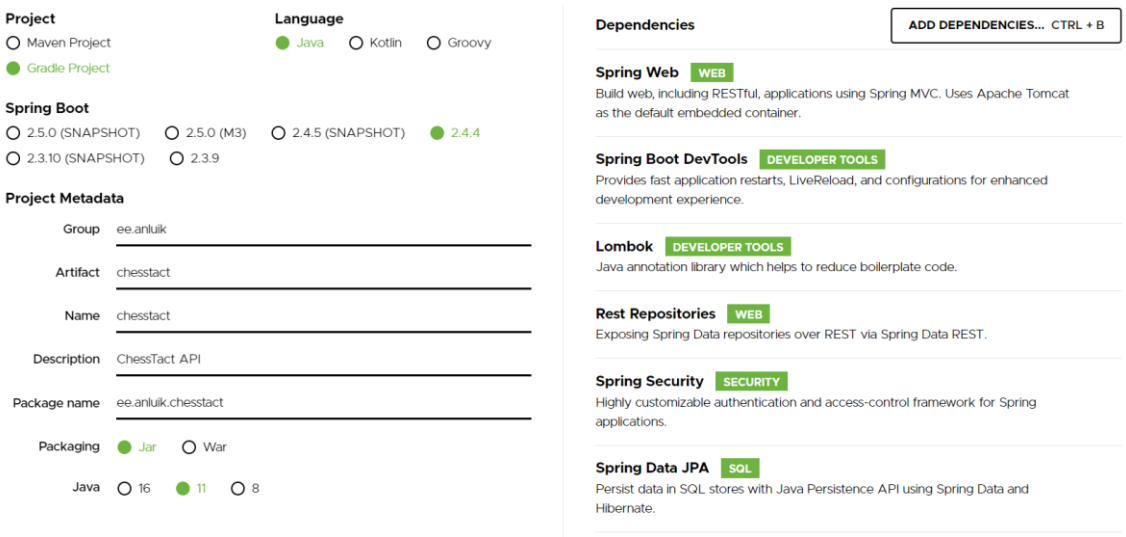
Järgnevas töö osas on kirjeldatud valminud rakenduse teostus. Arendusprotsess on jaotatud kahte peatükki: serveripoolse rakenduse arendus ja kliendipoolse rakenduse arendus.

5.1 Serveripoolse rakenduse arendus

Serveripoolne rakendus vastutab HTTP (*Hypettext Transfer Protocol*) päringute vastuvõtmise ja vastuste saatmiste eest. Samuti toimub serveri poolel andmete töötlus ning suhtlus andmebaasiga.

5.1.1 Spring projekti loomine

Rakenduse loomisel kasutati abitööriista Spring Initializr, mille abil genereeritakse arendaja eest projekti algstruktuur koos täpsustatud sõltuvustega. Joonisel 7 on näha rakenduse ehitamiseks valitud tööriistad ja sõltuvused koos inglise keelsete kirjeldustega.



Joonis 7. Spring Initializr'i abil projekti loomine koos sõltuvustega.

Tagarakenduse arenduseks kasutati Java 11 versiooni, mis on hiliseim pikaajalise toe (ing. k. *Long-Term Support*) väljaanne pärast Java 8-t. Pikaajaline tugi tähendab, et tarkvara väljaannet toetatakse pikema aja kestel kui tavaliselt. Sel ajal ei anta välja suuri funktsionaalseid täiendusi, mis rikuksid ühilduvust eelmiste versioonidega. Rakenduse

sõltuvuste haldamiseks kasutati Gradle tööriista, mis on avatud lähtekoodiga vahend Java projektide ehitamiseks ja haldamiseks.

Java rakenduste arendamisel on arendajal äärmiselt lai valik tehnoloogiatest, mille hulgast valida. Alates veebiraamistikest kuni ehitustööriistade ja logimistarkvardeni, mis kõik võib algajale arendajale olla ülekoormav. Spring Boot aga tegi projektiga alustamise kiireks ja lihtsaks. Vaja oli vaid märkida, millistel otstarvetel rakendust kasutama hakatakse ning automaatselt genereeriti täielikult arendamiseks valmis projekt, mida saab kergelt edasi konfigureerida vastavalt vajadustele.

5.1.2 Arhitektuuri ja struktuuri paika panek

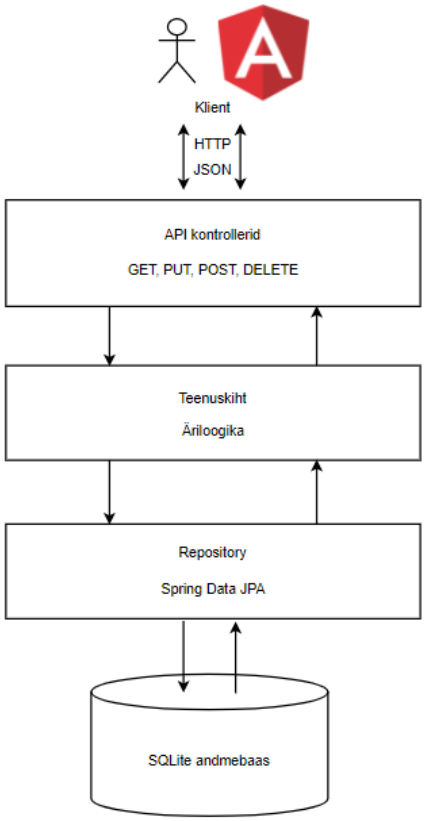
Tõhusa mitmekihilise arhitektuuri kujundamine on levinud viis tarbetute probleemide vältimiseks ja kvaliteetse koodi kirjutamiseks. Koodibaasi jagamine mitmeteks kihtideks vähendab hoolduseks vajalikke jõupingutusi, kuna igat kihti saab hallata teistest sõltumatult. Igal kihil on oma ülesanne, mis teeb rakenduse erinevad osad ning seosed nende vahel mõistetavateks ja kergesti hallatavateks [35].

Eelmainitu saavutamiseks on serveripoolse rakenduse koodibaas jaotatud kolme peamisesse kihti (Joonis 8):

1. API (*Application Programming Interface*) kontrollerite kiht on liides, läbi mille rakendus suhtleb välismaailmaga. API ehk rakendusliidese kihti süstitakse sisse teenuskihi meetodid.
2. Teenuskiht on äriloogikat sisaldav kiht, mis töötleb andmeid enne nende salvestamist andmebaasi või päringu tegijale väljastamist. Teenuskihti süstitakse repository kihi meetodid.
3. Repository on andmebaasiga suhtlev kiht.

Selline lahendus täidab seatud ootused rakendusele:

- vastu saab võtta päringuid ja saata välja vastuseid;
- andmebaasis saab talletada andmeid ja neid sealt kätte saada;
- enne päringu vastuse välja saatmist on võimalik andmeid vajadusel töödelda.

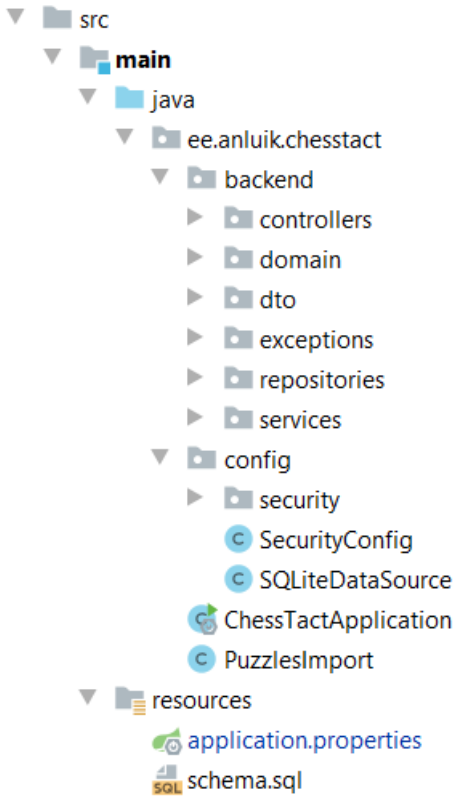


Joonis 8. Serveripoolse rakenduse arhitektuur.

Spring MVC tegi nende ootuste realiseerimise eriti kergelt saavutatavaks. Raamistikus eksisteerib kolme tüüpi tähtsaid komponente: kontrollereid, teenuseid ja hoidlaid. Nende vahel liikuvaid ressursse ehk andmeid esindab andmemudel. Iga komponendi klass on tähistatud vastavalt selle tüübile annotatsioonidega, kas *@RestController*, *@Service* või *@Repository*, mis koos moodustavad selle tüübi kihi. Kihid on iseseisvad, suheldes ainult vahetult enne või pärast teda asuva kihiga. Suhtlemisvõimaluse tagamiseks võtab raamistik suuresti vastutuse enda peale ning täpsustada oli vaja vaid sõltvuste süstimise (ing. k. *dependency injection*) metoodika abil, millised komponendid omavahel suhtleda tohivad.

Sellisel lähenemisel tõuseb esile üks miinus. Ühelt poolt arhitektuurimustrid ühendavad erinevaid raamistikke, andes neile ühise keele, kuidas kirjeldada nende võimekust. Teiselt poolt aga tekib olukord, kus arendajad teavad, milleks raamistik on võimeline, kuid ei pruugi olla teadlikud, kuidas ta seda tegelikult saavutab. Mõneti võib väita, et toimuks nagu midagi maagilist – vaja luua mõned klassid ja lisada paar annotatstiooni ning ongi töötav rakendus valmis.

Arhitektuuri kujundamisega samadel eesmärkidel on terve rakenduse kood struktureeritud ning sellest paistab välja ka arhitektuuri kihilisus (Joonis 9). Iga kiht on suletud ühte kindlasse paketti, mis on nimetatud esindatava kihi järgi. Lisaks mainitutele on olemas veel lisakihid, mis on hädavajalikud – neid kasutavad peamised kihid oma töö täitmiseks, näiteks veakäsitluseks ja andmete esitlemiseks või transpordiks.



Joonis 9. Serveripoolse rakenduse lõppstruktuur.

5.1.3 REST liides

Järgnevalt kirjeldatakse täpsemalt loodavat rakendusliidese kihti, mis vastutab päringute vastu võtmise ja vastuste välja saatmise eest. Suhtlus teiste rakendustega toimub läbi REST liidese.

REST on arhitektuuriline lahendus, mis seab paika mõned tavad, mida kasutatakse veebiteenuste arendamisel. Ühe tava järgi peaks URI (*Uniform Resource Identifier*) viitama ressursile, mitte toimingule, kus ressurss võib olla kas üksikobjekt (ing. k *singleton*) või kogum (ing. k. *collection*) [36].

Vastavalt ressursside nimedele identifitseeritakse liideses neid samanimeliste aadressidena, kuhu saab päringuid esitada. Näiteks aadressi /tournaments abil tuvastatakse võistluste kogumisressurssi ning üksikressursi pärimiseks kasutatakse

/tournaments/{tournament_id}. Järgnevalt on välja toodud kõik lubatud päringud loodud liidesele (Tabel 1).

Tabel 1. REST API võimalikud päringud.

Kirjeldus	**Meetod**	**Ressurss**
Registreeri võistlusele	POST	/participation
Võistluse tulemuste vaatamine	GET	/participation/{id}
Tulemuse salvestamine	PUT	/participation/{id}
Ühe mõistatuse vaatamine	GET	/puzzles/{id}
N mõistatuse pärimine raskusastmega DIFF	GET	/puzzles?N=&diff=
Võistluse vaatamine	GET	/tournaments/{id}
Eesolevate võistluste vaatamine. Parameeter USER abil saadakse teada, kas kasutaja on juba võistlusele registreeritud.	GET	/tournaments
Kasutaja võistluste vaatamine, millele ta on registreerinud.	GET	/tournaments/user
Võistluse loomine	POST	/tournaments
Kontole sisse logimine	GET	/login
Konto registreerimine	POST	/register

5.1.4 Turvalisus

Turvameetmeid pole enamasti mõtet ise teha, sest nii saavad nad tõenäoliselt sisaldama turvaauke, sest turvalisuse puhul on väga palju teemasid, mille peale ei pruugi osata mõtelda. Seepärast kasutatakse turvalisuse tagamiseks olemasolevat Spring Security raamistikku, mis on *de facto* standard Spring rakenduste kaitsmisel. Käesoleva lõputöö raames loodava rakenduse enamik vaateid on ligipääsetavad kõigile, olenemata, kas kasutaja on kontoga sisse loginud või mitte. Siiski seatud funktsionaalsete nõuete järgi saavad võistlusi luua ja nendel osaleda vaid registreerunud kasutajad. See tähendab, et ilma kontota kasutajatel on võimalik näha eesootavaid võistlusi, kuid nendele registreerimine ja uue võistluse loomise vormile ligipääs on keelatud.

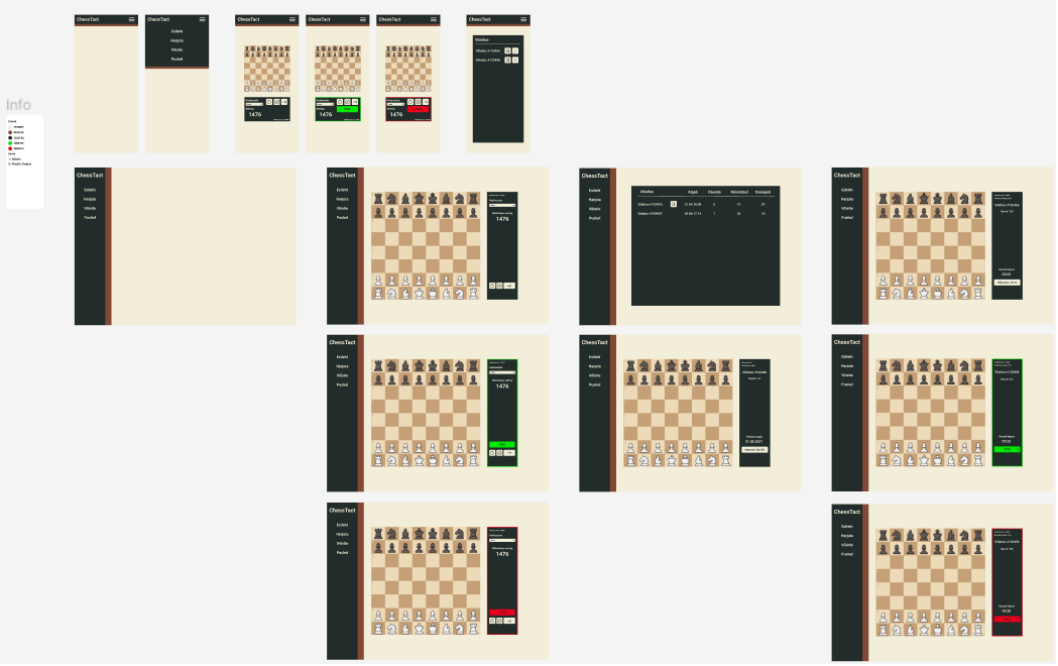
5.2 Kliendipoolse rakenduse arendus

Klientrakendusena loodi üheleherakendus, mille loomise tehnoloogiaks valiti Angulari veebirakenduste raamistik. Üheleherakenduste puhul tervet lehekülge kunagi mitu korda ei laeta – lehele navigeerides tõmbab brauser kogu rakenduse mällu, serveri poolelt käiakse JavaScripti abiga pärimas ainult andmeid.

Rakendusega alustamiseks võeti appi Angular CLI (*Command Line Interface*) , mis pakub mugavat ja kiiret viisi Angulari rakenduste initsialiseerimiseks, haldamiseks ja hooldamiseks. Käsuga „ng new“ luuakse uus projekt koos esialgse Angulari tavadele vastava failistruktuuriga, mille hulgas ka vajalikud konfiguratsioonifailid. Samuti installitakse vajalikud NPM teegid ja muud sõltuvused.

5.2.1 Rakenduse disain

Vastavalt analüüsis seatud nõuetele on teada funktsioonid, mis lõpptootes olemas olema peavad. Enne nende loogika arendamist loodi kavand, et teada, millised peaksid neid funktsioone täitvad komponendid välja nägema. Eesmärgiks võeti, et kuigi rakendus saab olema mõeldud enamasti kasutamiseks laua- ja sülearvutites, siis peaksid vaated kohanduma ka väiksemate ekraanidega. Kavandi järgi on hea hiljem kujundada ning vajadusel muudatusi teha. Esialgne kavand loodi Figma tarkvara abil (Joonis 10), mis oma laia tööriistade valikuga lubab disainida erinevate eesmärkidega prototüüpe.



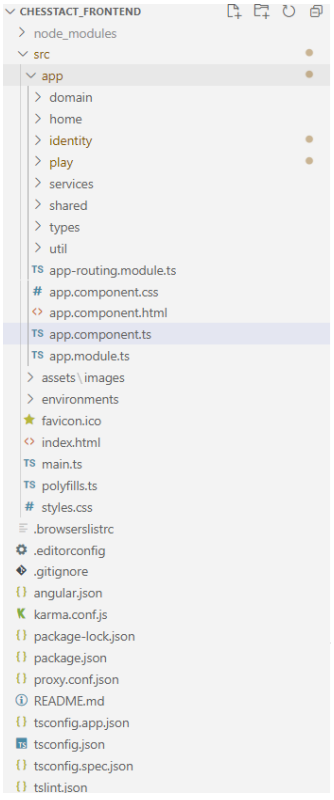
Joonis 10. Loodavate vaadete esialgne kavand Figmas.

Kavandi loomisel said paika vaadetes olevate komponentide stiilid ning ekraani suurusega kohanemise reeglid. Näiteks navigatsioonimenüü asendub väiksematel ekraanidel kõrvalmenüü asemel ekraani ülaservas asuva rippmenüüga, et teha rohkem ruumi malelaua kuvamiseks. Hiljem viidi menüü üle ekraani ülaserva, et võimalusel vältida komponentide ümberpaigutamist. Lisaks kohandub ekraani suurusega harjutamise ja võistlemise vaadetes malelaua seadistamiseks juhtmenüü. Kui ekraanil on ruumi piisavalt, siis asub menüü malelaua paremal küljel, kuid et kuvada malelauda ekraanil võimalikult suurena, liigutatakse see mõõtmete vähenedes hoopis laua alumisele küljele.

Rakenduse kasutamisel mobiilseadmetega võib siiski esineda ebamugavusi, kuid ligipääsetavad peavad olema kõik funktsioonid, mis on olemas suurematel seadmetel.

5.2.2 Funktsioonide loomine

Olenevalt vaadete ja nendes sisalduvate funktsioonide põhjal on nad jaotatud kaustadesse. Kogu eesrakenduse koodibaasi struktuur on nähtav joonisel 11.



Joonis 11. Eesrakenduse struktuur.

Kaustade genereerimiseks sai veelkord appi võtta Angulari käsurea. Käsk „ng generate component“ võimaldab täpsustatud asukohta luua komponendi kataloogi, mis sisaldab kahte TypeScripti faili (komponentklass ja testklass), ühte HTML ja ühte CSS faili. Need

täidetakse Angulari poolt koodimallidega, mis aitavad kiiresti asuda vaate ja sellega seotud loogika kirjutamise juurde.

Angular rakendus koosneb suuresti komponentidest ja veel teenustest, mis pakuvad rakendusele vajalikke väärtusi või funktsioone. Teenused muutuvad komponentidele kättesaadavaks sõltuvuste süstimise abil. Käesolevas töös kasutatakse neid mitmel eesmärgil:

- Serverile päringute tegemiseks, et andmeid saada või neid salvestada.
- Ligipääs teatud lehekülgedele võib mingite põhjuste tõttu olla kasutajale keelatud, näiteks uue võistluse loomise vorm kontota kasutajale või juba sisseloginud kasutajale sisselogimise vorm. Ligipääsu keelamiseks kasutati Angulari *valvur*-liideseid (ing. k. *guards*), mis kontrollivad enne lehekülje laadimist, kas toiminguks vajalikud tingimused on täidetud.
- Andmete eellaadimiseks, et enne vaate kuvamist kasutajale, oleks vajalikud andmed olemas. Vastasel juhul näeks kasutaja seni osaliselt laaditud lehte, kuni andmeid serverist päritakse. Neid liideseid nimetatakse *lahendajateks* (ing. k. *resolvers*).
- Väljaminevate taotluste ja saabuvate vastuste töötlemiseks on *pealtkuulaja*-liidesed (ing. k. *interceptors*). Antud projektis on üks selline teenus, mille otstarve on kasutaja poolt tehtud kindlaid GET päringuid meeles hoida ehk toimida vahemäluna. Näiteks teatud andmeid, nagu võistluste menüüs olevad kirjeid, hoitakse meeles, ning liikudes vahel teistele lehekülgedele ja tagasi, ei korrata varem serverile tehtud päringut, vaid andmed võetakse vahemälust.

6 Tulemused

Lõputöö eesmärgiks oli leida sobivaimad viisid probleemi lahendamiseks ja luua töötav ning nõuetele vastav rakendus. Püstitatud probleem sai lahendatud ning eesmärk täidetud.

Analüütilises osas teostati oluline olemasolevate rakenduste analüüs, mille ootamatu tulemus lükkas ümber lahendatava probleemi. Otsimise käigus leiti vastupidiselt probleemis väidetule 2 veebirakendust, mis lubasid selgitada välja suurema hulga osalejate seast valdkonna parima. Need rakendused aga ilmnesid otsingutulemustes välja ainult kindlate üksikute otsingufraaside- või sõnade puhul ning seda tihti alles sügaval tulemuste sees. Lisaks antud rakendused sisaldasid väljapaistvaid puudusi, mistõttu otsustati jätkata uue lahenduse loomisega.

Teostuse osas loodud lahendus vastab kõikidele seatud nõuetele ning on kasutatav nii mobiil- kui ka suurematel seadmetel (Joonis 12 ja Joonis 13).

Joonis 12. Võistlusvaade suurematel seadmetel.

Joonis 13. Harjutusvaade rakenduse avamisel mobiilseadmes.

Siiski rakendus valmis mõeldes just suurematele seadmetele ning seetõttu mobiilseadmetel kasutamisel võib vaadetes esineda ebamugavusi, kuid olemas on kõik funktsioonid.

Järgmiste sammudena oleks võimalik täiendada olemasolevaid funktsioone. Rakendus võiks kasvada huviliste jaoks teemakohaseks suhtlus- ja treenimisplatvormiks. Selleks saaks luua klubide, foorumite, edetabelite ning teiste mänguvariantide funktsioonid. Veelgi on mitmeid võimalusi kasutajamugavuse tõstmiseks. Hetkel on rakendusest puudu mõned omadused, mida kasutajad kindlasti tegelikult näha sooviksid. Näiteks jäi töö skoobist välja nõue, et võistluse ajal lehekülje värskendamisel peaks rakendus hoidma hetkeseisu meeles, kuid hetkel need andmed kaovad.

Valminud rakendust on võimalik näha ja proovida aadressil https://chesstact.itcollege.ee/. Soovitatav on seda avada kasutades Windows operatsioonisüsteemi ning Chrome või Edge veebibrauserit.

7 Kokkuvõte

Bakalaureusetöö raames loodi rakendus malemõistatuste harjutamiseks ja nendes võistlemiseks. Rakendus pakub võimalusi arendada male taktikalist mõtlemist ning selgitada välja osavaimad antud alal, milleks varasemalt leidus vähe ja/või kehvade funktsioonidega lahendusi.

Töös anti põhjalik ülevaade teema aktuaalsusest ning jätkati probleemiga tutvumist. Analüüsiti juba olemasolevaid sarnaseid rakendusi ning probleemist lähtuvalt pandi paika esialgsed minimaalsed nõuded. Analüüsi järgselt nähti vajadust uue rakenduse loomiseks ning alustati arendustöö planeerimisega. Selle käigus täiustati esialgseid nõudeid, koguti vajaminevad malemõistatused ning leiti abistavaid teeke töö lihtsustamiseks. Analüüsi viimase osana tehti põhjendatud tehnoloogilised valikud arendustööks.

Teostuse osas valmis töö alguses seatud eesmärki täitev rakendus. Realiseeriti kõik analüüsis paika seatud funktsionaalsed ja mittefunktsionaalsed nõuded. Nii kliendi- kui ka serveripoolse rakenduse loomisel jälgiti üldtuntud ja kasutatud raamistike puhul kehtivaid häid tavasid. Töö tulemuste peatükis on välja toodud rakenduse aadress.

Kasutatud kirjandus

[1] W. Farnsworth, Predator at the Chessboard: A Field Guide to Chess Tactics, 1. raamat, Lulu.com, 2011.

[2] P. Wong, „OzProblems,“ [Võrgumaterjal]. Available: https://www.ozproblems.com/problem-world/chess-problem. [Kasutatud 7. märts 2021].

[3] T. Genov, „Importance of Tactics part II,“ 31. august 2013. [Võrgumaterjal]. Available: https://www.chess.com/article/view/importance-of-tactics-part-ii. [Kasutatud 7. märts 2021].

[4] T. Genov, „Importance of Tactics,“ 18. august 2013. [Võrgumaterjal]. Available: https://www.chess.com/article/view/importance-pf-tactis. [Kasutatud 7. märts 2021].

[5] World Federation for Chess Composition, „Solving Competitions,“ [Võrgumaterjal]. Available: https://www.wfcc.ch/competitions/solving/. [Kasutatud 7. märts 2021].

[6] A. Steinbrink ja L. Palmans, „15th International Solving Contest (ISC),“ 2019. [Võrgumaterjal]. Available: https://www.wfcc.ch/competitions/solving/isc19/. [Kasutatud 7. märts 2021].

[7] Chess.com, „Chess Problems,“ [Võrgumaterjal]. Available: https://www.chess.com/terms/chess-problems#compositions. [Kasutatud 7. märts 2021].

[8] „Chess Puzzle,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Chess_puzzle. [Kasutatud 7. märts 2021].

[9] „Chess Problem,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Chess_problem. [Kasutatud 7. märts 2021].

[10] „Ray Robson Wins Puzzle Battle World Championship,“ 2021. [Võrgumaterjal]. Available: https://www.chess.com/news/view/ray-robson-wins-2021-puzzle-battle-championship. [Kasutatud 7. märts 2021].

[11] „8 Oldest Board Games in the World,“ Oldest.org, [Võrgumaterjal]. Available: https://www.oldest.org/entertainment/board-games/. [Kasutatud 8. märts 2021].

[12] C. Behler, „The 2020 Chess Boom,“ Medium, 2. detsember 2020. [Võrgumaterjal]. Available: https://medium.com/super-jump/the-2020-chess-boom-992427704a28. [Kasutatud 8. märts 2020].

[13] „Eesti Maleliit,“ Eesti Spordiregister, [Võrgumaterjal]. Available: https://www.spordiregister.ee/et/organisatsioon/4013/eesti_maleliit. [Kasutatud 7. märts 2021].

[14] „Eesti male kalenderplaan,“ Eesti Maleliit, [Võrgumaterjal]. Available: http://www.maleliit.ee/kalender/. [Kasutatud 7. märts 2021].

[15] V. Vorteil ja J. Laanpere, „Tarkvara arendusnõuded,“ Tallinna Ülikool, Tallinn.

[16] Google Inc., „Google Search,“ Google Inc., [Võrgumaterjal]. Available: https://www.google.com/. [Kasutatud 5. märts 2021].

[17] Google Inc., „Google Play,“ Google Inc., [Võrgumaterjal]. Available: https://play.google.com/store/apps. [Kasutatud 5. märts 2021].

[18] „lichess.org open database,“ [Võrgumaterjal]. Available: https://database.lichess.org/#puzzles. [Kasutatud 18. märts 2021].

[19] „npmjs,“ [Võrgumaterjal]. Available: https://www.npmjs.com/. [Kasutatud 18. märts 2021].

[20] „cm-chessboard,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/cm-chessboard. [Kasutatud 18. märts 2021].

[21] „chessground,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/chessground. [Kasutatud 18. märts 2021].

[22] „@lubert/chess.ts,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/@lubert/chess.ts. [Kasutatud 18. märts 2021].

[23] „chess,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/chess. [Kasutatud 18. märts 2021].

[24] „cm-chess,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/cm-chess. [Kasutatud 23. märts 2021].

[25] „chess.js,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/chess.js. [Kasutatud 23. märts 2020].

[26] „Spring vs. Spring Boot vs. Spring MVC,“ Javatpoint, [Võrgumaterjal]. Available: https://www.javatpoint.com/spring-vs-spring-boot-vs-spring-mvc. [Kasutatud 12. märts 2021].

[27] snyk.io, „JVM Ecosystem Report 2020,“ snyk.io, 2020.

[28] StackOverflow, „Stackoverflow Developer Survey 2020,“ StackOverflow, [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-web-frameworks. [Kasutatud 13. märts 2021].

[29] „Best Frontend Frameworks of 2021 for Web Development,“ SimForm, 5. jaanuar 2021. [Võrgumaterjal]. Available: https://www.simform.com/best-frontend-frameworks/. [Kasutatud 24. märts 2021].

[30] „8 Proven Reasons You Need Angular for Your Next Development Project,“ Grazitti Interactive, 5. juuni 2018. [Võrgumaterjal]. Available: https://www.grazitti.com/blog/8-proven-reasons-you-need-angular-for-your-next-development-project/. [Kasutatud 13. aprill 2021].

[31] „Angular Architecture Overview,“ Medium, 26. jaanuar 2019. [Võrgumaterjal]. Available: https://medium.com/@bhavikagarg8/angular-architecture-overview-1e7cc7483a0. [Kasutatud 13. aprill 2021].

[32] „DB-Engines Ranking,“ DB-Engines, March 2021. [Võrgumaterjal]. Available: https://db-engines.com/en/ranking. [Kasutatud 12. märts 2021].

[33] „SQLite is a Self Contained System,“ [Võrgumaterjal]. Available: https://www.sqlite.org/selfcontained.html. [Kasutatud 19. märts 2021].

[34] „SQLite is Serverless,“ [Võrgumaterjal]. Available: https://www.sqlite.org/serverless.html. [Kasutatud 19. märts 2021].

[35] A. Zanini, „Designing a Multi-Layered Architecture for Building RESTful Web Services With Spring Boot and Kotlin,“ Medium, 26. juuni 2020. [Võrgumaterjal]. Available: https://medium.com/swlh/designing-a-multi-layered-architecture-for-building-restful-web-services-with-spring-boot-and-a12ef85b77c9. [Kasutatud 22. aprill 2021].

[36] „REST Resource Naming Guide,“ Restfulapi, [Võrgumaterjal]. Available: https://restfulapi.net/resource-naming/. [Kasutatud 16. aprill 2021].

[37] „JWT,“ jwt.io, [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 16. aprill 2021].

Lisa 1 – Lihtlitsents

Mina, Andri Luik

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Rakendus malemõistatuste harjutamiseks ja võistluste korraldamiseks", mille juhendaja on Märt Kalmo.
   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.

Lisa 2 – Olemasolevate rakenduste otsimiseks kasutatud otsingusõnad ja -fraasid

Eesti keeles:

- maleprobleemid
- malemõistatused
- malemõistatuste turniir
- malemõistatuste võistlus

Inglise keeles:

- solve chess online
- chess solving
- chess problems
- chess puzzles
- chess challenges
- chess tactics
- chess problems tournament
- chess puzzles competition

Lisa 3 – Olemasolevate rakenduste analüüsis kasutatud rakendused

Veebirakendused

chess.com
chesspuzzle.net
tactics.chessbase.com
chesshub.com
sparkchess.com
chesstempo.com
chesspuzzles.com
365chess.com
apronus.com
lichess.org
thechesswebsite.com
kidchess.com
chesskid.com
chessmatec.com
chess24.com
ideachess.com
gameknot.com
chessity.com
jchess.net
chessebook.com
chess-steps.eu
chessvis.com
chessproblem.my-free-games.com
chesspuzzlesonline.com
justchess.biz
ozproblems.com
cariboutests.com/games/chess.php
usefulchess.com
chessproblems.org
wtharvey.com
blitztactics.com
chessanytime.com
chesscup.org
chess.cool

Mobiilirakendused

Chess Kings
RealChess
SzachMaks
Tactics Frenzy
Chess Tactics for Beginners
Chess Puzzles – Simple & Fun
iChess
Chess Problems, tactics, puzzles
Fun Chess Problems Free
Chess Tactics Pro
Chess Puzzles

Lisa 4 – Malelauateegi testimiseks kasutatud kood Angular komponendis

```
@Component({
  templateUrl: './play.component.html',
  styleUrls: ['./play.component.css']
})
export class PlayComponent implements OnInit {
  // cannot use import... Because of the hyphen in 'cm-chessboard'?
  board = require('src/cm-chessboard/Chessboard.js');
  INPUT_EVENT_TYPE = this.board.INPUT_EVENT_TYPE;

  ngOnInit(): void {
    const chess = new this.board.Chessboard(
      document.getElementById("chessboard"),
      {position: "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR"});

    chess.enableMoveInput((event: any) => {
        switch (event.type) {
            case this.INPUT_EVENT_TYPE.moveStart:
                return true
            case this.INPUT_EVENT_TYPE.moveDone:
                return true
            case this.INPUT_EVENT_TYPE.moveCanceled:
                return false;
            default:
              return false;
        }
    }, this.board.COLOR.white)
  }
}
```