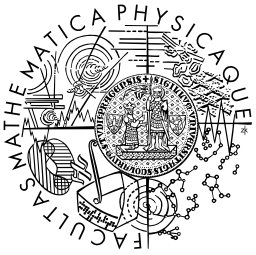


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Josef Sezemský

Systém pro správu šachových soutěží

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Filip Kliber

Studijní program: Informatika

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chtěl bych poděkovat panu Mgr. Filipu Kliberovi za vedení této práce a poskytnutí dobrých rad, kdykoliv jsem potřeboval.

Název práce: Systém pro správu šachových soutěží

Autor: Josef Sezemský

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Filip Kliber, Katedra distribuovaných a spolehlivých systémů

Abstrakt: Žijeme v době, kdy probíhá digitalizace ve všech odvětvích. Přesto pro organizátory šachových turnajů ani jejich účastníky neexistuje moderní software, který by uspokojil jejich požadavky. V této práci provedeme analýzu problémů, které souvisí s organizací šachového turnaje. Současně též zanalyzujeme programy, které jsou v současnosti na řešení této problematiky používány. Dle výsledků analýzy navrhneme a vytvoříme webovou aplikaci, která usnadní průběh šachového turnaje všem zúčastněným.

Klíčová slova: systém, šachy, turnaj, soutěž, web, aplikace

Title: System for management of chess competitions

Author: Josef Sezemský

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Filip Kliber, Department of Distributed and Dependable System

Abstract: We live in a time when digitalization is taking place in all sectors. Yet for chess tournament organisers and participants, there is no modern software to satisfy their requirements. In this thesis we will analyze the individual steps related to the organization of a chess tournament. At the same time, we will also analyze the programs that are currently used to tackle these difficulties. According to the results of the analysis, we will design and develop a web application that will facilitate the chess tournament for all participants.

Keywords: system, chess, tournament, competition, web, application

Obsah

Úvod

Během šachových turnajů a soutěží můžeme zpozorovat určité nedostatky v efektivitě organizace. Touto prací bychom měli ulehčit práci organizátorům menších lokálních turnajů. Program by měl usnadnit všechny části organizace turnaje a automatizovat je, aby organizátor pouze zadával data a program za něj vykonal vše ostatní.

Zásadní problém nalézáme v nedostatku kvalitních programů pro kompletní organizaci turnaje. Na jednotlivé části turnaje jsou používány různé technologie, které většinou nejsou na danou činnost uzpůsobeny. Například registrace na turnaje probíhá nejčastěji pomocí e-mailové komunikace. Tímto může vzniknout několik problémů z důvodu nekonzistence dat, jelikož je vše závislé na pečlivosti organizátora, kterému současný stav přidělává mnoho práce navíc, která ho odvádí od dalších organizačních činností. Například již během přihlašování na turnaj pomocí zmíněné e-mailové komunikace, existuje možnost, že nedojde ke správnému porozumění mezi organizátorem a hráčem, čímž může dojít ke špatné identifikaci hráče. Po přihlášení uživatelů musí organizátor opět ručně zadávat data do losovacího nástroje, čímž může u složitých jmen dojít k jejich záměně. Těchto problémů by se šlo lehce vyvarovat, kdyby existoval software, který by tyto kroky provázal do jednoho komplexního systému a tím minimalizoval chybu lidského faktoru.

Cílem práce je vytvořit program, pomocí kterého organizátoři menších lokálních turnajů budou moci uspořádat turnaj bez toho, aniž by se museli zabývat technickými detaily. Cílem není vytvořit program, který by měl nahradit profesionální losovací nástroje pro mistrovské turnaje, ale vytvořit aplikaci, která bude snadno pochopitelná i pro nezkušené organizátory. Budeme se snažit implementovat pouze ty nejdůležitější funkcionality, aby bylo dosaženo jednoduché ovladatelnosti.

V následujících kapitolách je možné nalézt informace o průběhu šachového turnaje, výhody a nevýhody v současnosti používaných nástrojů, analýzu technologického řešení pro implementaci aplikace, programátorskou dokumentaci a též uživatelskou dokumentaci k výsledné aplikaci.

1 Jak probíhá šachový turnaj

1.1 Části šachového turnaje

Před samotnou analýzou problému a jeho následným vyřešením je nutné si uvědomit, jaké části má šachový turnaj a jak spolu souvisí. Tyto informace nám pomohou lépe navrhnout řešení, které bude splňovat všechny funkční požadavky a v budoucnu umožní dalším vývojářům funkcionality upravit či vylepšit.

1.1.1 Vytvoření turnaje - propozice

Aby účastníci věděli, zda je pro ně turnaj vhodný, je nutné jim poskytnout určité informace. Informace o turnaji jsou poskytnuty většinou ve formě dokumentu. Tomuto dokumentu se říká propozice. Propozice by měly obsahovat všechny potřebné informace o turnaji, například adresu turnaje, způsob losování jednotlivých kol, datum konání a další. Propozice mohou být poté zveřejněny na webových stránkách klubu pořadatele, webových stránkách šachového svazu či jiné platformě.

1.1.2 Registrace

Pokud se účastník po přečtení propozic rozhodne pro účast v turnaji, případně se účasnit jako hráč týmu, je nutné navázat kontakt s organizátorem. Dle požadavků mu poté poskytnout osobní informace, aby organizátor věděl, kdo se bude turnaje účastnit a dle toho mohl uzpůsobit organizaci. Dle rozhodnutí organizátora může být registrace umožněna i na místě v den konání turnaje.

1.1.3 Losování kol

Po zahájení turnaje je nutné v každém kole každému účastníkovi přiřadit protihráče, proti kterému bude hrát a bude mít možnost získat body. Přiřazení protihráče probíhá dle předem definovaného způsobu. Výběr losovacího formátu náleží organizátorovi turnaje a je na jeho uvážení, který formát je pro daný turnaj nejvhodnější dle prostorových možností, počtu účastníků a dalších faktorů. O těchto formátech je možné se podrobněji dočíst v dalších podkapitolách.

1.1.4 Výsledky

Po odehrání všech kol turnaje již zbývá pouze vyhodnotit konečné pořadí. Dle zvoleného losování může být tento proces odlišný. Výsledky je poté vhodné sdílet s ostatními účastníky, aby byly dostupné i po skončení turnaje i pro nezúčastněné osoby.

1.2 Druhy turnajů dle rozlosování

Dle způsobu párování účastníků můžeme dělit turnaje na tři druhy:

- *Round Robin*
- *PlayOff*
- *Švýcarský systém*

Každý z těchto formátů má jisté výhody i nevýhody, které si nyní představíme.

1.2.1 Round robin

Round Robin, každý s každým, je druh turnajového párování, během kterého bude každý hráč postupně hrát zápas proti všem ostatním účastníkům turnaje. Mezi hlavní přednosti patří jeho spravedlivost. Zápas proběhne mezi každou možnou dvojicí a tudíž nikdo nemůže skončit na horší pozici, jelikož hrál proti těžším protihráčům, což může nastat například ve *švýcarském systému*. Na druhou stranu tento systém vykazuje jednu hlavní nevýhodu — velký počet kol, které je nutné odehrát. Počet kol je minimálně o jedna menší než počet hráčů. Z tohoto důvodu je tento druh turnaje od určitého počtu hráčů velmi nepraktický. Dá se říci, že se hodí na menší uzavřené turnaje, které je možné v případě nutnosti hrát během více dnů. Případně ho lze využít i na turnaje s kratší časovou kontrolou, jelikož může během jednoho dne dojít k odehrání většího počtu kol.

1.2.2 PlayOff

V kontrastu rozřazování *Round Robin* se nachází *PlayOff* turnaje. Do dalšího kola postupují vždy pouze vítězové zápasů předchozího kola. Tento způsob zaručuje rychlé ukončení turnaje, jelikož počet kol nutných ke zjištění vítěze roste logaritmicky s počtem účastníků. Vhodný je tedy na turnaje o značném počtu hráčů. Jelikož je ale hráč v případě prohry automaticky vyřazen a v dalších kolech již nehraje, je vhodné tento formát použít pouze v případě, kdy nám počet hráčů či velikost prostor neumožňuje jinak. Dalším problémem je nemožnost určení konkrétního pořadí dalších účastníků, kteří se poté nachází na dělených místech. Kvůli těmto nevýhodám se na šachových turnajích, s výjimkou mistrovských soutěží, nevyskytuje. Účastníci amatérských turnajů si chtějí především užít šachovou hru a vyřazení po prvním kole by způsobovala mnohá zklamání.

1.2.3 Švýcarský systém

Většina šachových soutěží se losuje *švýcarským systémem* nazývaného též jako „*Swiss*“, jelikož jeho výhody převažují nad jeho nevýhodami. Jeho hlavní myšlenka spočívá v párování hráčů se stejným počtem bodů, což z něj dělá velmi univerzální a variabilní způsob párování pro většinu turnajů. Jelikož se v turnaji neodehrají zápasy mezi hráči s výrazným výkonnostním rozdílem, dojde k odehrání všech důležitých zápasů během několika kol a též dojde ke spolehlivému určení celkového umístění všech hráčů. Abychom s jistotou určili vítěze a nenastala nám situace, kdy dva hráči budou mít maximální možný počet bodů bez odehraného vzájemného

zápasu, je nutné odehrát stejný počet kol jako v *PlayOff* turnaji. Jelikož *švýcarský systém* vyžaduje odehrát v každém kole počet zápasů odpovídající polovině počtu hráčů, je nutné mít adekvátní prostory, proto není tolik využívaný v ostatních sportech. Jako příklad můžeme uvést tenis, kde bychom pro každé kolo turnaje potřebovali velké množství tenisových kurtů. Naopak během šachových turnajů nám vystačuje pouze šachovnice s figurkami, která je oproti tenisovému kurtu velmi levná. Jelikož *švýcarský systém* umožňuje všem hráčům odehrát celý turnaj a nutný počet kol roste pouze logaritmicky s počtem hráčů, je pro šachové turnaje upřednostňován před ostatními typy losování.

Proces párování závisí na jeho konkrétní variaci. Je na daném softwaru, jak jej implementuje, ale je vhodné, aby dodržoval základní pokyny od FIDE[1].[1] Následující odstavce popíší hlavní myšlenky, jak párování probíhá.[2]

Pokud se turnaje účastní lichý počet hráčů, tak mezi hráče doplníme imaginární osobu, která prohraje každý zápas a slouží pouze k usnadnění procesu párování. Při procesu párování roztřídíme hráče do skupin, dle celkového počtu bodů získaných v předchozích kolech. V případě, že se jedná o kolo první, má každý hráč nula bodů a tudíž jsou všichni hráči ve stejné skupině. Skupiny seřadíme sestupně dle počtu bodů každého účastníka ve skupině.

Hráče z první skupiny rozdělíme na dvě stejně velké poloviny. V případě lichého počtu hráčů ve skupině, bude první polovina o jednoho hráče větší. Nalezneme párování, které bude obsahovat co nejvíce možných párů, které spolu ještě v turnaji nehrály. V každém páru musí ale být jeden hráč z první poloviny a druhý hráč z druhé poloviny. Zbývající hráče, kterým nebylo možné najít soupeře, posuneme do další skupiny. Pokračujeme stejným způsobem při párování další skupiny.

V případě, že dojde k posunu nenulového počtu hráčů do další skupiny, je rozdělení na dvě poloviny jiné. V tomto případě jsou do první poloviny dáni hráči, kteří byli přesunuti z předchozí skupiny. Do druhé poloviny jsou dáni hráči, kteří se ve skupině již nacházeli.

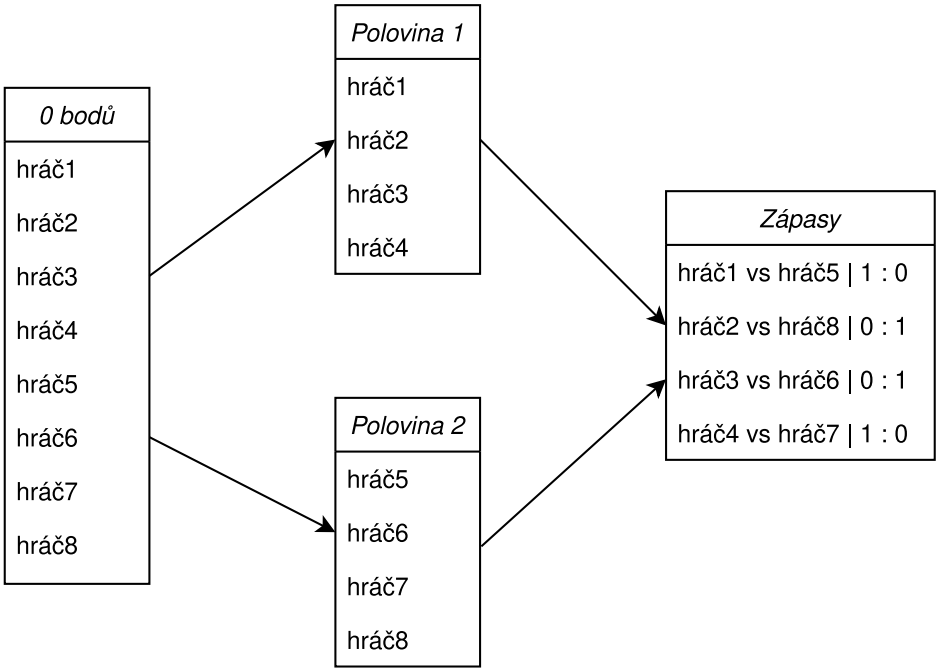
Tento proces je opakován, dokud každý z hráčů nemá svého protihráče.

V pozdějších kolech může dojít k situaci, kdy nám v poslední skupině zůstanou pouze hráči, kteří spolu v předchozích kolech již hráli. V tomto případě zrušíme páry z předchozí skupiny a hráče z těchto párů společně se zbývajícími hráči, kteří nemají soupeře, použijeme k novému pokusu. Pokusíme se najít párování, aby každý hráč měl protihráče. Pokud se nám to nepovede zrušíme opět páry v další skupině a pokusíme se opět najít párování. Takto pokračujeme dokud se nám nepovede nalézt korektní párování. Může nastat situace, kdy ani zrušení párů všech skupin nepovede ke korektnímu párování všech hráčů. V tomto případě nelze párování vygenerovat a je na organizátorovi, jak tuto situaci vyřeší.

Abychom lépe pochopili, jak proces párování *švýcarským systémem* probíhá, popíšeme si jej na příkladu. Uvažme turnaj, do kterého se přihlásilo osm hráčů se jmény *hráč1, hráč2, hráč3...* Začneme párováním prvního kola. Jelikož hráči ještě neodehráli žádný zápas, mají všichni nula bodů a tím pádem budou všichni patřit do jedné skupiny. Skupinu se všemi osmi hráči rozdělíme na dvě stejně velké poloviny. V první skupině se nachází *hráč1, hráč2, hráč3, hráč4*. Ve druhé skupině se nachází zbývající hráči. Nyní je potřeba nalézt, co nejlepší párování mezi těmito dvěma skupinami. Nejlepší párování je to, které bude obsahovat co

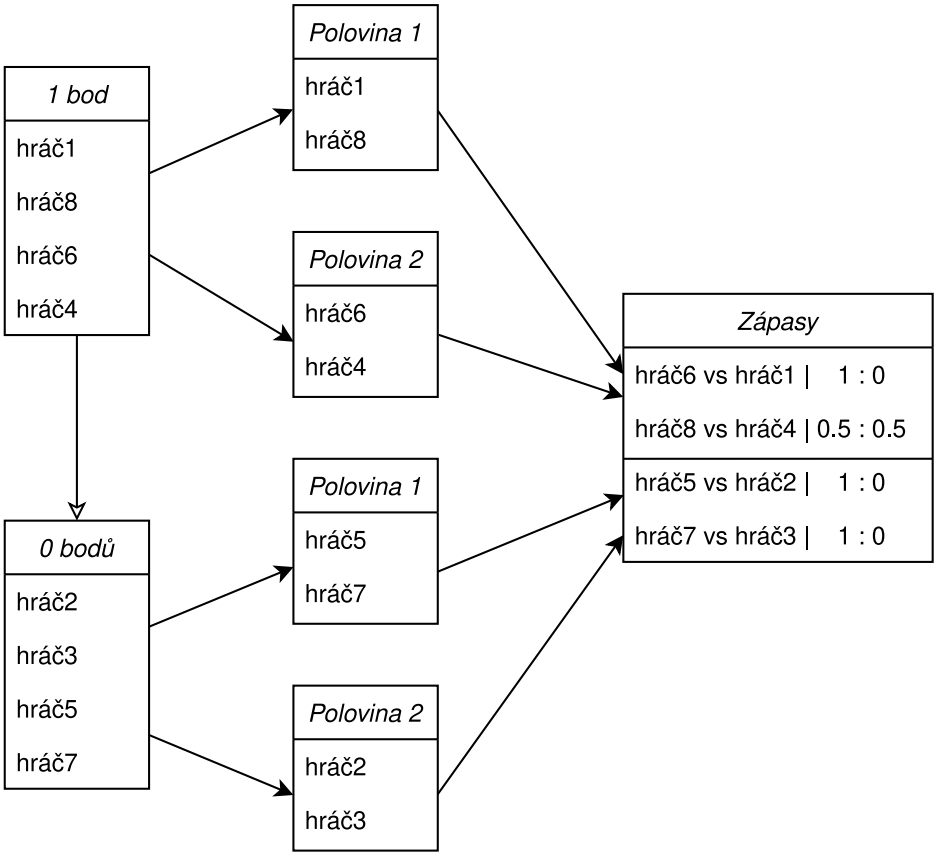
[1]Mezinárodní organizace, která sdružuje šachové svazy jednotlivých zemí. Organizuje a spravuje všechny mistrovské turnaje.

nejvíce dvojic. Jelikož párujeme první kolo, nemusíme zatím dbát na to, aby spolu nehráli hráči, kteří proti sobě v turnaji již hráli. Nejlepší párování, kdy nalezneme protihráče každému hráči, je například *hráč1 vs hráč5, hráč2 vs hráč8, hráč3 vs hráč6, hráč4 vs hráč7*. Tito hráči by proti sobě odehráli zápasy v nichž by získali určité body. Pro naši ukázku budeme předpokládat, že první a čtvrtý zápas skončily vítězstvím prvního jmenovaného hráče, druhý a třetí zápas skončily vítězstvím druhého jmenovaného hráče.



Obrázek 1.1 Švýcarský systém - první kolo

Pokračovat budeme druhým kolem. Jelikož v předchozím kole neskončil žádný zápas remízou, máme pouze dvě skupiny. V první skupině se nacházejí hráči s jedním bodem a ve druhé skupině se nacházejí hráči s nula body. Seřadíme skupiny dle počtu bodů hráčů, což je v tomto případě jednoduché, první bude skupina s hráči s jedním bodem a druhá bude skupina s hráči s nula body. První skupinu rozdělíme na dvě poloviny. Poloviny v našem případě budou *hráč1, hráč8* a *hráč6, hráč4*. Opět nalezneme párování s co nejvíce možnými páry, což v tomto případě odpovídá například *hráč6 vs hráč1, hráč8 vs hráč4*. Povedlo se nám spárovat všechny hráče ze skupiny, tudíž do další skupiny nebudeme přesouvat žádného hráče. Pokračovat budeme druhou skupinou. Opět rozdělíme hráče na dvě poloviny, například *hráč5, hráč7* a *hráč2, hráč3*. Mezi těmito polovinami nalezneme nejlepší párování. To je například *hráč5 vs hráč2, hráč7 vs hráč3*. Ve skupině nám nezbyl žádný nespárovaný hráč a jelikož se jednalo o poslední skupinu, máme párování pro druhé kolo. Pro náš příklad budeme uvažovat výsledky zápasů, *hráč6, hráč5* a *hráč7* svůj zápas vyhráli, *hráč8* a *hráč4* spolu remizovali.



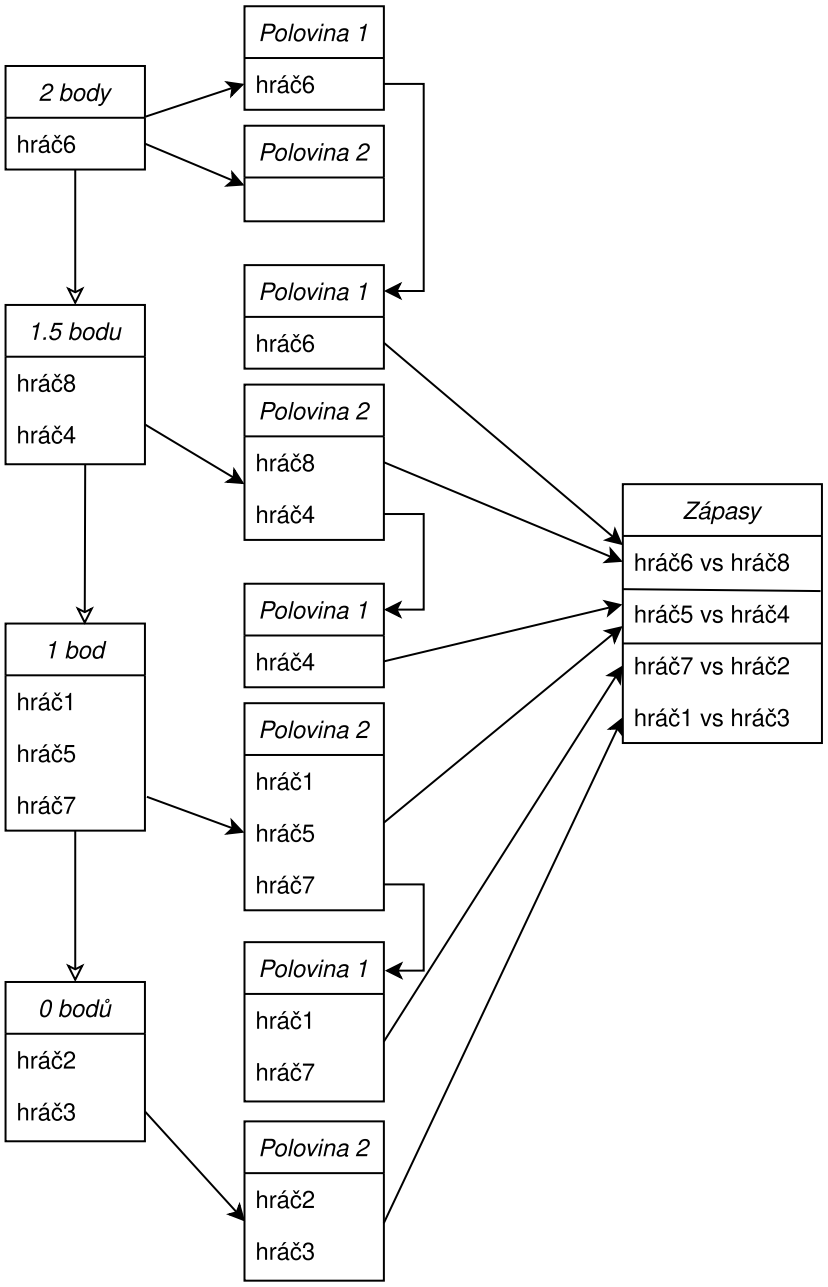
Obrázek 1.2 Švýcarský systém - druhé kolo

Pokračovat budeme třetím a tedy i posledním kolem. Opět si hráče rozdělíme do skupin podle získaných bodů v předchozích kolech.

- 2 body - *hráč6*
- 1,5 bodu - *hráč8, hráč4*
- 1 bod - *hráč1, hráč5, hráč7*
- 0 bodů - *hráč2, hráč3*

Začneme párovací proces. Seřadíme sestupně skupiny podle počtu bodů. Rozdělíme první skupinu na dvě poloviny. Jelikož skupina obsahuje pouze jednoho hráče, bude tento hráč v první polovině a ve druhé polovině nikdo nebude. Pokusíme se nalézt co nejlepší párování. V tomto případě nelze nikoho spárovat a hráče tedy posuneme do další skupiny. Pokračovat budeme skupinou s hráči, kteří mají jeden a půl bodu. Jelikož do této skupiny jsme přesunuli hráče *hráč6*, bude rozdělení na poloviny jiné. Do první poloviny dáme přesunutého hráče a do druhé poloviny dáme hráče, kteří se v této skupině již nacházeli. Nejlepší párování je například *hráč6 vs hráč8*, více než jeden pár zde nemůžeme vytvořit. Ve skupině nám zbyl nespárovaný *hráč4*, kterého přesuneme do další skupiny. Pokračovat budeme skupinou s hráči s jedním bodem. Jelikož do této skupiny jsme přesunuli hráče *hráč4*, provedeme rozdělení do polovin jako v předchozím případě. V první polovině bude

hráč4 a ve druhé polovině budou hráči *hráč1, hráč5, hráč7*. Nalezneme nejlepší párování. V tomto případě můžeme opět vytvořit pouze jeden pár a to je například *hráč5 vs hráč4*. Zbývající hráče *hráč1, hráč7* přesuneme do další skupiny. Nyní už nám zbývá pouze poslední skupina a to jest skupina s hráči, kteří mají nula bodů. Jelikož jsme opět přesunuli nenulový počet hráčů, tak do první poloviny dáme přesunuté hráče a do druhé hráče, kteří v této skupině již byli. Párování, kde vytvoříme maximální dva možné páry, je například *hráč7 vs hráč2* a *hráč1 vs hráč3*. Nezbyl nám žádný hráč a tudíž máme párování pro třetí kolo hotové. Jelikož již nebudeme pokračovat dalším kolem, výsledky jsou pro nás již nepodstatné a posloužili by pouze k určení konečného výsledku.



Obrázek 1.3 Švýcarský systém - třetí kolo

Z tohoto příkladu by již nyní mělo být zřejmé, jak párování probíhá.

Pokud by při určování konečného pořadí zůstalo několik hráčů se stejným bodovým ziskem na děleném místě, použijí se k vyřešení této situace tzv. pomocná hodnocení. O prioritě použití rozhoduje organizátor turnaje. V příručce pro turnaje od FIDE [3] nalezneme například tato pomocná hodnocení:

- **Buchholz** - odpovídá součtu bodů všech soupeřů, se kterými daný hráč odehrál partii
- **Krácený Buchholz** - odpovídá klasickému Buchholz, od kterého odečteme počet bodů nejsilnějšího a nejslabšího soupeře (nejsilnější a nejslabší soupeř jsou určeni dle pořadí v turnaji)
- **SonnenBorn-Berger** - součet bodů všech soupeřů, které jsou přenásobené bodovým ziskem hráče v partii s daným soupeřem
- **Počet partií s černými figurami** - hrát s černými figurami značí jistou nevýhodu, tudíž upřednostníme hráče, který hrál s touto nevýhodou vícekrát

1.3 Druhy turnajů dle účastníků

Turnaje dle účastníků dělíme na dva typy a to na turnaje pro jednotlivce a na týmové turnaje. V turnajích jednotlivců dochází k párování účastníků, kteří proti sobě budou hrát v daném kole. V týmových turnajích dojde k párování týmů, které proti sobě budou hrát. Poté dojde ke spárování jednotlivých hráčů daných týmů do podzápasů. Výsledek zápasu je poté určen jako součet bodů, které získali jednotliví hráči týmu v jejich zápasech.

1.4 Druhy turnajů dle časové kontroly

Organizace FIDE rozeznává tři druhy časových kontrol, turnaj můžeme tedy označit jednou z těchto možností: *Blitz, Rapid* nebo *Classical*. Přiřazení turnaje do dané skupiny závisí na časové kontrole daného turnaje. $PocatecniCas$ určuje jaký základní čas dostane každý hráčů. $PridavekZaTah$ určuje počet vteřin, které se přičtou ke zbývajícímu času po provedení tahu.

Pro *Blitz* turnaje platí:

$$PocatecniCas + 60 * PridavekZaTah < 15\,minut$$

Pro *Rapid* turnaje platí:

$$15\,minut <= PocatecniCas + 60 * PridavekZaTah < 60\,minut$$

Jako *Classical* označíme turnaje nesplňující ani jednu z těchto podmínek.[4]

2 Technologie využívané k organizaci turnajů v současnosti

Chtěli bychom, aby náš software vylepšil současný stav digitalizace organizace šachových turnajů. V této kapitole provedeme analýzu programů, které jsou v současnosti využívány. Získané informace poté využijeme při implementaci programu, který eliminuje jejich problémy a zároveň se inspiruje z jejich dobrých vlastností.

2.1 Přihlášení na turnaj

Po diskuzi se spoluhráči z šachového oddílu jsme dospěli k závěru, že na přihlášení do turnaje se používá převážně e-mailová komunikace, případně komunikace telefonická. Výjimečně se lze setkat s využitím webových formulářů. V těchto případech ale dochází k repetitivnímu vyplňování jednotlivých údajů, což vede k častým chybám z důvodu překlepů a nedorozumění. V případě nutnosti odhlášení se z turnaje je nutné opět kontaktovat organizátora, identifikovat se, a organizátor poté ve své evidenci provede odhlášení. Náš program by měl tento problém vyřešit a zároveň zjednodušit celý proces přihlášení i odhlášení sebe či týmu z turnaje. V ideálním případě by organizátor nemusel do celého procesu registrace vůbec zasahovat.

2.2 Losovací nástroje

Pro losování turnaje je vhodné použít software určený k losování turnaje předem daným způsobem. Turnaje losované stylem *PlayOff* jsou zvládnutelné i bez pomocného programu. Je vhodné ho ale využít, abychom neudělali chybu z nepozornosti. Naopak turnaje typu *Swiss* je velmi náročné losovat bez pomocného softwaru z důvodu jeho náročných pravidel, které je nutné při losování dodržet. Nyní si ukážeme v současnosti běžně dostupné programy a identifikujeme jejich výhody a nevýhody. Získané informace nám poté pomohou při implementaci naší aplikace.

2.2.1 Swiss-Manager

Swiss-Manager je jeden z nejznámějších programu na losování turnajů švýcarským systémem, který poskytuje velké množství modifikací párování. Nicméně to s sebou přináší řadu výhod i nevýhod. Program umožňuje nastavit vše do nejmenších detailů, což se využívá při mistrovských turnajích. Nezkušené uživatele ale mohou všechny možnosti snadno zahltit. Mohou působit natolik matoucím dojmem, že nejsou schopni v programu provádět základní činnosti.

Po jeho spuštění si můžeme okamžitě všimnout jeho neintuitivního uživatelského rozhraní. Začínající uživatel neví, jak by měl program ovládat, aby vůbec provedl základní úkony jako je registrace hráče nebo rozlosování jednotlivých kol. Je zřejmé, že po čase dojde k zapamatování těchto akcí, ale začátky s tímto

programem mohou být velmi náročné. Navíc ikony tlačítek provádějící dané úkony jsou velmi malých rozměrů. Osoby se sníženými zrakovými schopnostmi nemusí být schopni rozlišit jednotlivá tlačítka a program se pro ně stane neovladatelným.

Další nevýhodou je možnost užívání programu pouze po zakoupení licence za nemalou částku 150 EUR, či 75 EUR v případě Light verze [5]. Pro organizátory malých lokálních turnaj není tato částka zanedbatelná a může spoustu lidí od organizace turnaje odradit.

Na výčet nevýhod můžeme též zařadit nutnost instalace programu, což pro organizátory, kteří nemusí být technicky zdatní, může představovat zásadní problém.

Program je ale celkově kvalitní pro zkušené organizátory velkých turnajů, kteří využijí veškeré funkcionality. Nicméně pro běžného uživatele ovládání představuje podstatnou výzvu.[6]

2.2.2 SwissSys

Desktopovou alternativou k programu Swiss-Manager může představovat SwissSys, u kterého můžeme objevit stejné nedostatky jako u programu Swiss-Manager. Pro užívání je potřeba si zakoupit licenci za 99 amerických dolarů[7] a uživatel na první pohled nedokáže odhadnout jak provádět základní činnosti. Též je nutná instalace programu.[8]

2.2.3 Swips

Jako příklad webové aplikace můžeme uvést aplikaci Swips, která umožňuje losování turnajů *švýcarským systémem*. Ovládaní je již více intuitivní než u předchozích příkladů, nevýhodou ale stále zůstává nutnost zaplacení poplatku za možnost používání pokročilejších funkcí, jako je například nastavení pomocných hodnocení či vyšší počet hráčů. Dále není možné s tímto programem uspořádat týmový turnaj.[9]

2.2.4 Free-swiss

Free-swiss představuje bezplatnou aplikaci na losování turnaje švýcarským systémem. Ke stažení je nutná registrace na stránce, která kombinuje angličtinu s němčinou. Dle obrázků na oficiální stránce umožňuje registrovat hráče a vygenerovat párování hráčů pro jednotlivá kola. Ostatní informace nebylo možné ověřit, jelikož během registrace ani po několika pokusech nedorazil potvrzovací e-mail. Ten je potřebný pro stažení programu a nebylo tedy možné program vyzkoušet. Dle dostupné dokumentace můžeme konstatovat, že uživatelské rozhraní velmi odrazuje program používat.[10]

2.2.5 Ostatní open-source nástroje

Další běžně dostupné nástroje, které nevyžadují poplatek za jejich používání, nejsou pro běžné uživatele prakticky použitelné. Ve většině případů je potřeba pro úspěšnou instalaci znalost balíčkovacích systémů a nebo chybí uživatelské rozhraní. Jelikož od organizátorů nemůžeme očekávat znalost balíčkovacích systémů a terminálu, je použití těchto programů vyloučeno.

2.2.6 Shrnutí

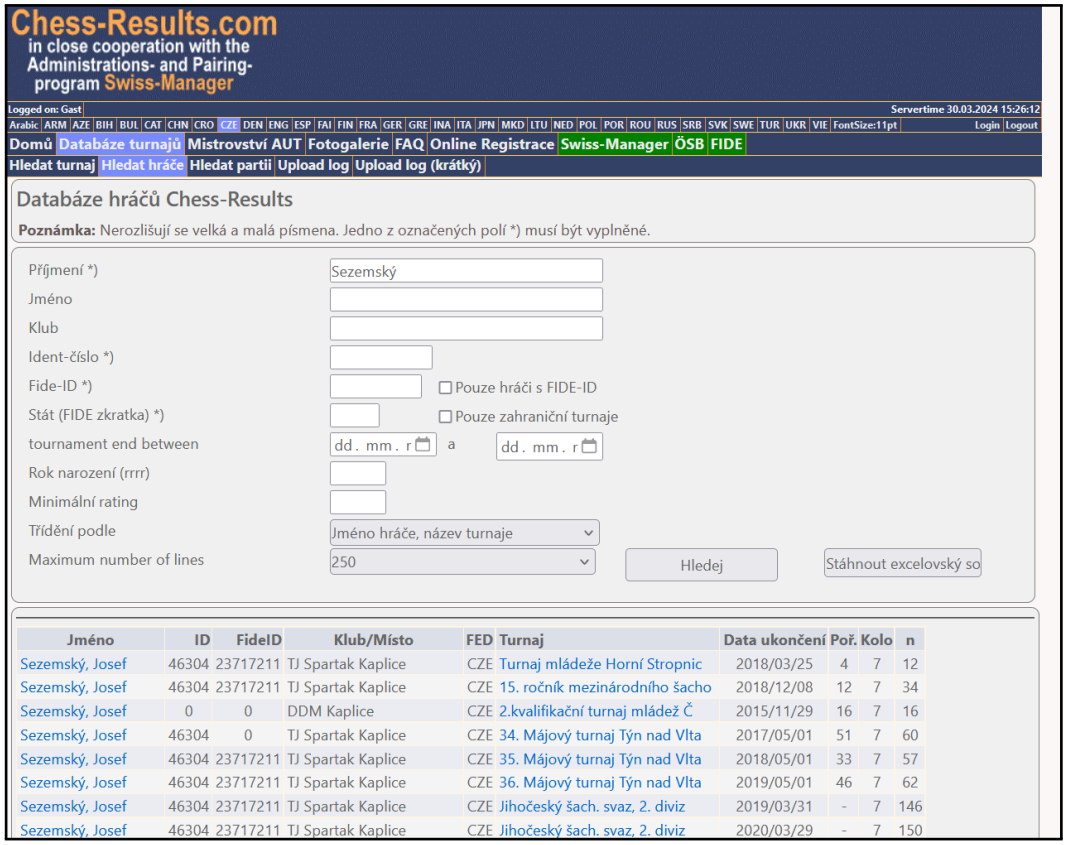
Dle těchto příkladů můžeme vypozorovat, že hlavní nevýhody, které tyto programy mají, je jejich užívání až po zakoupení licence, neintuitivní uživatelské rozhraní, případně chybějící funkce bezplatné verze. Tyto programy též slouží pouze na losování turnaje a neumožňují další funkcionality jako je například přihlášení a odhlášení z turnaje. Pokud bychom chtěli využít open-source variant programů, je vyžadována znalost ovládání terminálu, což od organizátorů šachových turnajů nemůžeme očekávat. Na dané problémy se zaměříme a pokusíme se je vyřešit v naší aplikaci.

2.3 Zveřejnění výsledků

Pro nahrávaní výsledků nebyla nalezena platforma, která by se univerzálně zaměřovala na šachové turnaje. Jednotlivé losovací nástroje ale většinou nabízejí své vlastní řešení.

Výsledky turnajů používající již zmíněný losovací nástroj Swiss-manager jsou dostupné na webové stránce, která zobrazuje výsledky spravované právě tímto programem[1]. Stránka ale již nesplňuje moderní parametry a uživateli neposkytuje vše, co by od ní očekával. Její hlavní nedostatek spočívá ve vyhledávání turnajů určitého hráče. Pokud bychom chtěli zjistit, zda nedošlo k aktualizaci výsledků turnaje, musíme mít uložený odkaz na daný turnaj nebo ho neustále složitě vyhledávat přes vyhledávací formulář. Problém nabývá na závažnosti, pokud takových turnajů sledujeme více. Hráči též musí spoléhat na pečlivost organizátora, zda vyplní důležité informace. Jinak může nastat situace, kdy není možné rozlišit, zda se v různých turnajích jedná o stejného hráče (viz 2.1, řádky s turnaji číslo 2, 3, 4). Dalšího problému si všimneme při otevření webové stránky na dipleji s vysokým nebo naopak nízkým rozlišením, velikost stránky v pixelech zůstává pořád stejná. Též chybí responzivní rozhraní pro mobilní zařízení.

[1] chess-results.com



Obrázek 2.1 Hledání turnajů určitého hráče na stránce chess-results.com s nedokonalým vyplněním celé obrazovky na pravé straně

Program SwissSys nabízí též vlastní webovou stránku s výsledky[2]. Má ale stejné problémy jako webová stránka s výsledky pro program Swiss-manager. Vyjmenované problémy jsou ale ještě závažnější.

Swips nám poskytuje výsledky na webové stránce, která splňuje moderní standardy.[3] Nenachází se zde závažnější nedostatky, které se týkají rozložení uživatelského rozhraní, a může nám sloužit jako vzor. Nedostatek ale opět nacházíme ve složitém vyhledávání oblíbených turnajů.

Po analýze těchto příkladů můžeme konstatovat, že hlavními problémy jsou nedokonalé uživatelské rozhraní, nemožnost automatického zobrazování preferovaných informací a nemožnost zjištění totožnosti osoby z turnaje.

[2] swisssys.com/event_selector.php
[3] swips.eu/en/tournaments

3 Analýza implementace

3.1 Hlavní funkcionality

Než uděláme rozhodnutí o technologiích, které použijeme při implementaci, si rozmyslíme hlavní činnosti, které by organizátoři a hráči chtěli v aplikaci dělat. Od toho se bude poté odvíjet způsob ukládání dat a návrh celé aplikace.

3.1.1 Hráč

- Přihlášení své osoby na turnaj a případné odhlášení z turnaje
- Vytvoření týmu a jeho přihlášení na turnaj
- Zjištění soupeře pro dané kolo
- Zobrazení svých zápasů v turnaji
- Zjištění finálních výsledků
- Prohlížení výsledků jednotlivých hráčů na turnajích

3.1.2 Organizátor

- Zobrazení registrovaných hráčů nebo týmů
- Vytvoření nového turnaje
- Vygenerování rozlosování kola
- Vygenerování a zobrazení výsledků

3.2 Platforma

V prvé řadě je nutné rozhodnout o platformě, pro kterou bude aplikace vyvíjena. Při pohledu na potřebné funkcionality vidíme opakující se požadavek na manipulaci s daty a jejich následné zobrazení. Jednotlivé akce jsou tedy nárazového charakteru a nedochází k souběžnému náročnému výpočtu po delší dobu. Data by měla být přístupná pro všechny uživatele, budeme tedy potřebovat, aby byla uložena na dostupném úložišti. Pro snížení reakční doby aplikace a výpočetní zátěže bychom chtěli minimalizovat množství přenášených dat.

Pokud by data byla uložena na úložišti dostupném výhradně přes internet, mohl by nastat problém v případě neexistujícího internetového připojení na místě turnaje. Jelikož se ale šachové turnaje odehrávají v budovách a prakticky v každé místnosti se v současnosti nachází internetové připojení, není tento typ úložiště překážkou.

Mezi organizátory může patřit široká škála osob, tedy i lidé staršího věku. Proto by uspořádání turnaje bylo mělo být co nejjednodušší z důvodu slabší počítačové gramotnosti ve starších věkových kategoriích.

Z těchto důvodů byla zvolena webová aplikace, jelikož data mohou být uložena na serveru společně s aplikací, což sníží reakční dobu. Webová aplikace též uživatelům umožní okamžité užívání aplikace bez nutnosti její instalace. V případě nové verze aplikace nebude nutné zatěžovat uživatele aktualizacemi a oni se budou moci soustředit pouze na organizaci a hraní turnajů. Zároveň bude naše aplikace fungovat na všech operačních systémech, které mají moderní webový prohlížeč. Jelikož předpokládáme, že všichni hráči mají během turnajů u sebe jen mobilní telefon a nikoliv notebook, odpadne nám nutnost vývoje samostatné mobilní aplikace, aby i hráči měli přístup k aplikaci během turnaje.

3.3 Programovací jazyk a framework

Abychom se odprostili od řešení technických detailů, které vývoj webové aplikace přináší, je důležité vybrat vhodný programovací jazyk a případný framework, který nám usnadní implementaci projektu.

3.3.1 PHP

PHP patří mezi standardní programovací jazyk na straně serveru. Tento jazyk ale vznikl již v roce 1995 a za tu dobu prošel web mnoha změnami.[11] Během těchto téměř třiceti let došlo k vytvoření nových datových formátů a programovacích standardů, pro které již existuje lepší podpora v moderních jazycích. Též nepatří mezi silně typované jazyky, což ve velkém projektu v pokročilejších fázích může přinášet zbytečné chyby způsobené nepozorností. Další problém tkví v omezené komunikaci mezi serverovou částí aplikace a částí běžící v prohlížeči uživatele.

3.3.2 JavaScript

Další možnost nalezneme v JavaScript frameworkách jako je například React či Vue.js. Dle srovnání popularity od Jakuba Svachy a Artura Kulpy během posledních deseti vývoj webových aplikací probíhá právě v nich.[12] Znamená to tedy větší povědomí o těchto frameworkách mezi vývojáři, což by usnadnilo rozšíření aplikace dalšími lidmi. Vývoj těchto frameworků po dobu několika let znamená jejich vyspělost a podporu mnoha různorodých činností. Jejich rozšířenost znamená i lepší integraci s ostatními technologiemi. Umožňují také vhodnou komunikaci mezi serverovou a klientskou částí aplikace.

3.3.3 Blazor

Mezi relativně nové technologie patří framework pro vývoj webových aplikací od firmy Microsoft.[13] Tento framework běží na platformě .NET, která nám poskytuje velké množství knihoven pro prakticky jakoukoliv činnost. Též nám umožňuje využít programovací jazyk C# jak na klientské, tak serverové části. Dalším důležitým aspektem je integrovaná podpora různých pokročilých funkcí jako je stream rendering či podpora autentifikace a autorizace v konkrétních částech aplikace. I přestože JavaScript frameworky by mohly představovat lepší volbu z důvodu jejich vyspělosti a rozšířenosti, byl pro vývoj naší aplikace vybrán právě Blazor. Za frameworkem Blazor totiž stojí společnost Microsoft a jelikož

je to velká společnost a Blazor staví na základech ASP.NET, můžeme věřit ve větší potenciál Blazoru do budoucna. Výběr byl také zčásti ovlivněn osobními preferencemi pro jazyk C#.

3.3.4 Frontend

Přestože moderní HTML a CSS, které se používají v Blazor frameworku na frontend části aplikace, nám umožňují vyvinout profesionálně vypadající webovou aplikaci, mají jisté nedostatky. Během vývoje musíme upravovat jednotlivé detaily každého elementu, abychom dosáhli cílené podoby. Řešení můžeme nalézt v různých knihovnách s již předvytvořenými styly. Jako hlavního zástupce můžeme uvést knihovnu Bootstrap. Bootstrap by nám umožnil efektivní vývoj frontend části aplikace s určitými výhodami ležící například v automatické responzivnosti nebo moderně vypadajících stylech. Jelikož ale bude aplikace napsaná pomocí Blazor frameworku, jeví se jako řešení použít navíc knihovnu napsanou přímo pro Blazor, která by využila jeho plný potenciál. Použijeme tedy knihovnu Blazor Bootstrap, což je knihovna využívající výhod Bootstrapu a zároveň Blazoru. Jednotlivé elementy jsou napsány jako Blazor komponenty s již aplikovanými styly základního Bootstrapu. Dojde tedy k eliminaci syntaktických chyb při psaní stylů, jelikož vše bude kontrolováno překladačem jazyka. Dojde tedy k podstatnému zvýšení efektivity práce z důvodu využití silné typovanosti jazyka C#.

3.4 Způsob ukládání dat

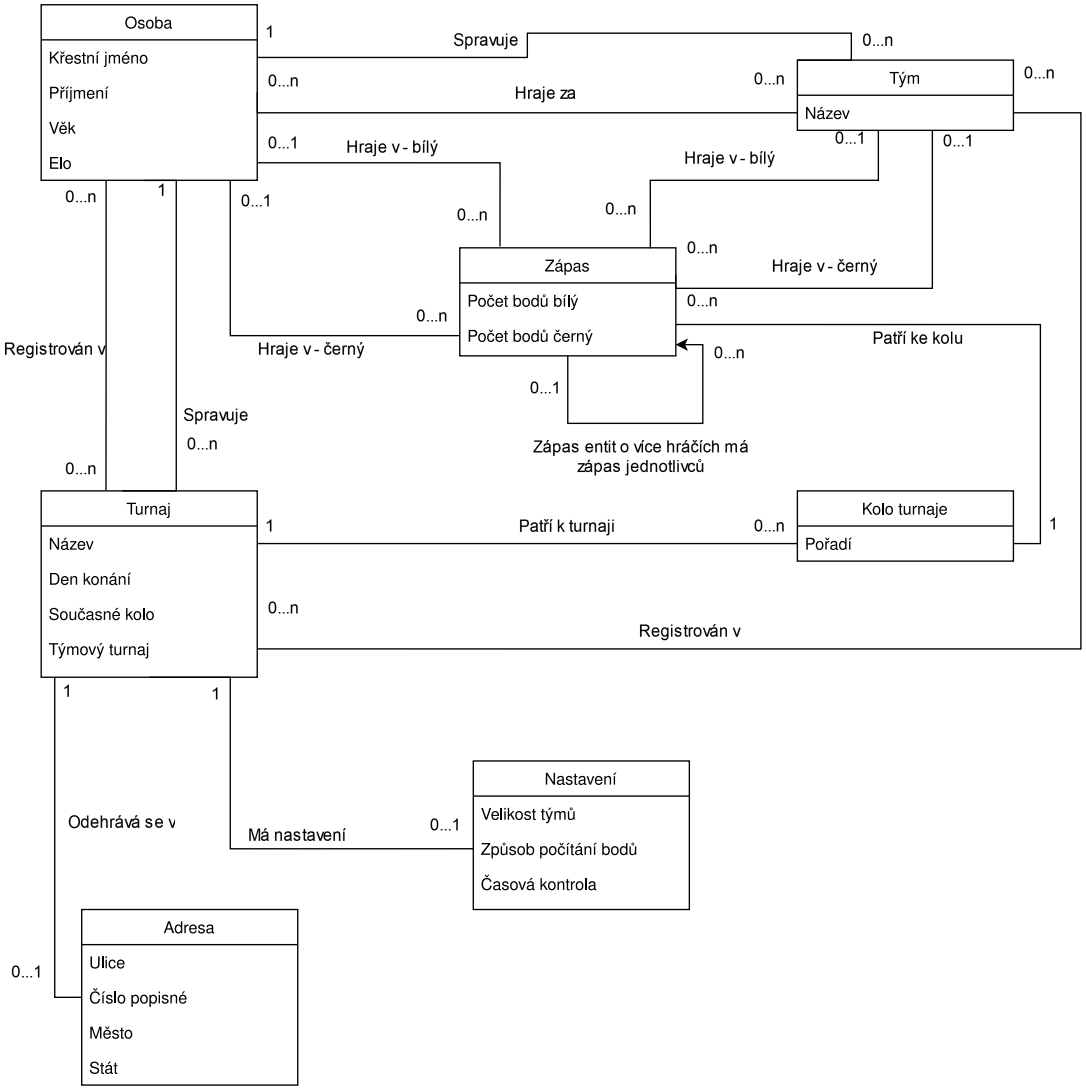
Pro dlouhodobé uložení dat a možnosti nad nimi efektivně provádět dotazy i s nimi manipulovat, je vhodné využít konkrétní možnosti z řad databázových systémů.

Před výběrem databázového systému si rozmyslíme, jaké údaje vlastně budeme chtít ukládat. Ukládané údaje a jejich formát velmi ovlivní naše rozhodnutí o výběru databázového systému.

- Údaje o hráči
- Jednotlivé hráče týmu
- Adresa hrací místnosti turnaje
- Jak bude turnaj losován a další organizační záležitosti
- Registrované hráče na turnaji
- Registrované týmy na turnaji
- Rozlosování pro všechna kola
- Výsledky zápasů
- Výsledky jednotlivých zápasů jednotlivců v případě týmového turnaje
- Nastavení daného turnaje

- Kdo může upravovat turnaj
- Kdo může upravovat tým

Z těchto údajů můžeme vypozorovat pár hlavních entit - Hráč/Osoba, Tým, Turnaj, Kolo turnaje, Zápas, Nastavení turnaje, Adresa.



Obrázek 3.1 UML diagram

Nad těmito daty budou prováděny dotazy na získání informací, které uživatelé budou potřebovat. Dle informací v předešlých kapitolách můžeme vypozorovat, že budeme především používat dotazy na získání konkrétních entit, nebudeme provádět agregační dotazy.

3.4.1 Odhad velikosti dat

Ve FIDE bylo k roku 2019 registrováno přes 170 000 aktivních hráčů [14]. Abychom si vytvořili horní odhad, budeme předpokládat, že všechny turnaje

budou spravovány naším programem. Pokud bychom uvážili, že se každý hráč účastní deseti turnajů ročně a každého turnaje se účastní průměrně padesát hráčů. Získáme informaci, že se ve světě odehraje přibližně 34 000 turnajů. Pro každý turnaj budeme uvažovat průměrně deset kol. To znamená, že do databáze bude muset být každý rok uloženo přes 340 000 kol s tím, že každé kolo bude mít dvacet pět zápasů. Celkový počet zápasů za rok by tedy odpovídal hodnotě 8 500 000. Tento odhad je samozřejmě velmi nadhodnocený a reálná hodnota bude mnohem nižší. I tyto maximální hodnoty nejsou ale nijak závratně velké a výkon databáze nebude hrát při výběru zásadní roli. Při výběru bychom se tedy měli zaměřit na ostatní faktory.

3.4.2 Výběr databázového systému

Diskuzi o výběru databázového systému začneme relačními databázemi, které vznikly již ve 20. století a jsou to tedy robustní prověřené systémy, které umožňují ukládat data v tabulkách a velmi dobře udržovat jejich konzistenci. Pokud se ale pozorněji podíváme na UML diagram, vidíme, že většina entit má malé množství vlastností, které bychom chtěli ukládat. Naopak vidíme značné množství propojení a vazeb typu M:N, které by v databázi způsobovaly nutnost vytvářet vazební tabulky. Data jsou také variabilní. Například v tabulce se zápasy jsou zápasy týmů a jednotlivců, ale v budoucnu by mohly přibýt i zápasy klubů, což by opět zkomplikovalo datový model. Nesmíme ale opomenout výhodu relačních databází, která spočívá v udržování konzistence dat. Pokud ale vezmeme v potaz zmíněné nevýhody u našeho konkrétního případu se zmíněnou výhodou, nepředstavují relační databáze ideálního kandidáta pro ukládání dat. Budeme tedy hledat dále v sekci nerelačních databází. Začneme databázemi dokumentovými. Tento druh nemá jasně dané schéma a je schopen pracovat s velmi variabilními daty. Dokumentové databáze ale nejsou vhodné pro data s vazbami typu M:N, což naše data obsahují. Tento typ se tedy také nejeví jako ideální kandidát. Zvážit můžeme i databáze typu klíč-hodnota, které přestože poskytují flexibilní schéma, vysokou rychlost a škálovatelnost, se jeví jako velmi nevhodný kandidát z důvodu neexistující podpory vztahů mezi daty. Naše aplikace bude obsahovat velké množství těchto vztahů, tudíž tuto možnost zamítáme. Pokračovat budeme grafovými databázemi. Hlavní výhodu nalezneme v hranách, kterými můžeme reprezentovat vztahy mezi daty. Dle UML diagramu komplexita těchto propojení není triviální a grafová databáze by nám umožnila tyto informace efektivně ukládat. Slabé stránky grafové databáze spočívají v agregačních dotazech, které ale téměř nebudeme využívat. Naopak bude vynikat v dotazech, které vyžadují procházení jednotlivých entit po propojeních, což bude zahrnovat většinu dotazů. Též databáze nevyžaduje striktní specifikaci datového modelu, což nám umožní snadněji ukládat variabilní data. Dále se zaměříme na další typ nerelačních databází – sloupcové databáze. Ty umožňují efektivní provádění dotazů nad sloupci, což bychom využili například při provádění analýzy dat jednotlivých údajů. V našem programu ale analýzu dat prakticky nevyužijeme. Výhodu můžeme také nalézt v efektivitě ukládání dat, které poté zabírají méně místa na disku, což znamená rychlejší provádění dotazu. Sloupcové databáze ale mají podobný problém jako databáze relační - ukládání je prováděno do tabulek, což by v našem případě znamenalo komplikovanější datový model.[15]

3.4.3 Konečný výběr databázového systému

Po této diskuzi nad jednotlivými druhy se jako nejvhodnější pro naší aplikaci jeví grafová databáze z důvodu velkého množství propojení entit a snadného zachycení komplexních informací, které by nebyly snadné uložit v databázových systémech s fixním datovým modelem. Jako konkrétní systém byl zvolen *Neo4j*, se kterým bude snadné komunikovat z důvodu podpory v podobě oficiální knihovny pro jazyk C# od tvůrců tohoto databázového systému. Zároveň pro mnohaletý vývoj a rozšířenost umožní bezproblémový vývoj aplikace.

3.5 Uživatelské rozhraní

Aby náš program využívalo co nejvíce lidí a práce s programem byla přívětivá, nesmíme opomenout navržení stylu, jakým bude vytvářeno uživatelské rozhraní. Je potřeba si opět uvědomit, jak vypadá složení potencionálních uživatelů. Organizovat turnaje budou především dospělé osoby, které nemusí mít dostatečnou počítačovou gramotnost a též mohou být sníženy jejich zrakové schopnosti. Zobrazený obsah v uživatelském rozhraní by tedy měl být co nejjednodušší s minimem rušivých elementů. Pro snížení zmatení uživatele, by každá akce měly poskytovat zpětnou vizuální vazbu. Jednotlivé druhy zpráv i akcí by měly být barevně rozlišeny, aby uživatel vždy věděl, zda výsledek jeho akce byl úspěšný či nikoliv. Jelikož se naše aplikace zaměřuje i na nezkušené organizátory, měli bychom vždy zvážit, zda přidaná funkcionalita nesníží ovladatelnost aplikace a uživatelé by nebyli schopni novou funkcionalitu správně využívat.

3.6 Problém přístupu více uživatelům ke stejným datům

Jelikož program bude využívat více uživatelů najednou, je nutné vyřešit situaci, kdy dojde ke změně dat jiným uživatelem. Pokud by jeden uživatel změnil data v databázi, ostatní uživatelé by mohli tyto změny přepsat, jelikož by ve své aplikaci měli stále načtená neaktuální data. Pokud si ale uvědomíme, kdo může spravovat určité údaje, zjistíme, že synchronizace dat nepředstavuje závažný problém.

Turnaj je vždy spravován pouze jedním uživatelem, tudíž ostatní uživatelé nemohou data o turnaji přepsat. Aktuální data o turnaji se jim poté vždy zobrazí po opětovném načtení stránky. Aktualizovaná data o turnaji není nutné zobrazovat okamžitě, tudíž časová prodleva nepředstavuje problém. Profil uživatele je též spravován pouze jedním uživatelem a tudíž se opět nemusíme starat o synchronizaci jako v předešlém případě. Stejná situace nastává se správcem týmu. Jedinými akcemi, které by mohly způsobit problém v případě neaktuálních dat, jsou přihlášení a odhlášení z turnaje. Před provedením akce související s přihlášením a odhlášením musíme nejprve ověřit, zda nedošlo k aktualizaci dat. Pokud by například uživatel měl načtenou přihlašovací stránku do turnaje a přihlášení mu bylo v danou chvíli umožněno, nic mu nebrání mít stránku otevřenou i několik dní. Poté už by mohla být kapacita turnaje naplněna, ale uživatel by se stále mohl na turnaj přihlásit. V ostatních případech nedochází k souběžné manipulaci se stejnými daty.

4 Programátorská dokumentace

V této části popíšeme strukturu naší aplikace a způsob jejího nasazení v produkčním prostředí. Popis se bude zabývat pouze nejdůležitějšími elementy, nebude tedy zacházet do detailů, které jsou uchopitelné z kódu. Doplňující informace je možné vyčíst z dokumentačních komentářů.

4.1 Architektura

Jak jsme si již rozmysleli, většina operací v aplikaci jednorázově mění nebo získává data z databáze a zobrazuje je uživateli. Z tohoto důvodu je celá aplikace rozdělena do čtyř vrstev, kde každá z nich má jasně definované činnosti. Vrstva může vždy komunikovat pouze s vrstvami o jednu úroveň výš nebo naopak o jednu úroveň níž.

- **Model** - přímá komunikace s databázovým systémem, mapování dat z databáze, spouštění dotazů
- **Presenter** - skládání základních metod provádějících manipulaci dat v databázi do komplexnějších operací
- **Komunikační komponenta** - používá metody z *Presenteru* k načtení a odesílání dat, poskytuje předpočítaná data k zobrazení
- **Zobrazovací komponenta** - má k dispozici již všechna potřebná data, zobrazuje data uživateli

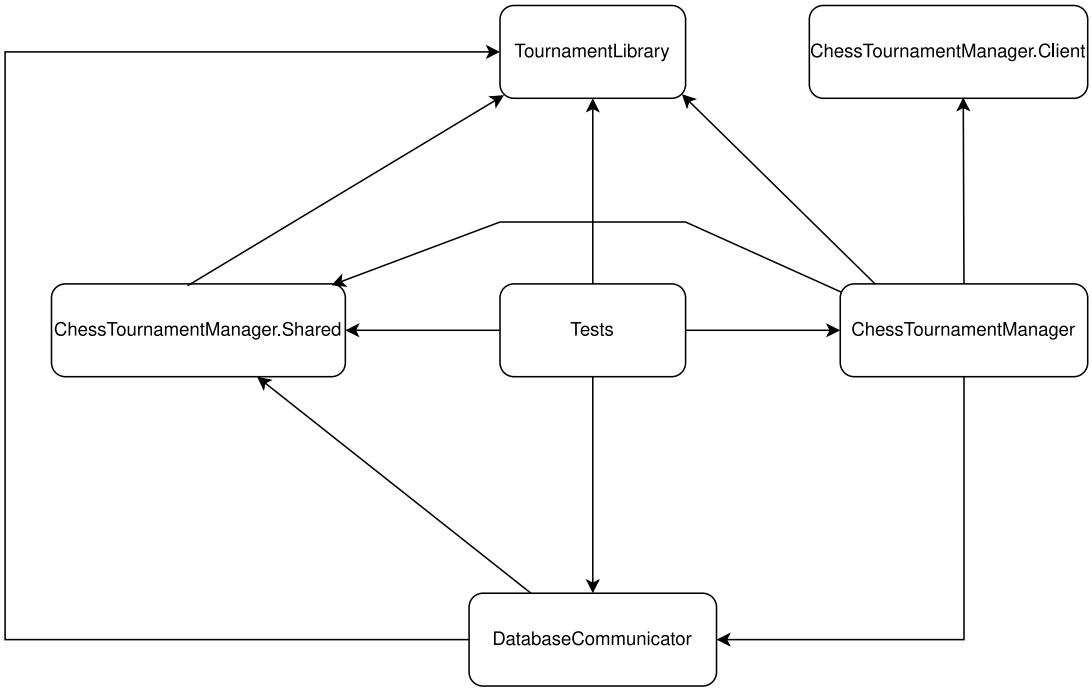
Abychom při používání služeb v naší aplikaci nemuseli pokaždé ručně vytvářet instanci konkrétní služby se všemi závislostmi, využíváme dependency injection kontejneru. Do něj registrujeme všechny využívané služby. Instanci služby získáme v komponentě pomocí klíčového slova `@inject`.

4.2 Projekty

Aplikace se skládá z několika projektů, kde každý z nich má jasně definované funkcionality. Další rozšíření jsou poté vždy implementována do projektu kam dle funkčnosti spadají.

- **TournamentLibrary** - Knihovna ve které nalezneme všechny třídy na generování kol a výsledků. Knihovna je dále využívána v ostatním projektech.
- **DatabaseCommunicator** - Reprezentuje vrstvu *Model*. Provádí veškeré dotazy nad databází a provádí mappping dat.
- **Tests** - Obsahuje veškeré implementované testy pro ostatní projekty.
- **ChessTournamentManager.Client** - Je určen pro funkcionality běžící v prohlížeči uživatele. V naší aplikaci ale není využíván. Je zde ponechán pouze pro případné využití těchto funkcionalit při dalším vývoji.

- **ChessTournamentManager.Shared** - Obsahuje různé základní funkcionality, které naleznou využití v ostatních projektech. Jedná se o jednoduché funkcionality, které se často používají.
- **ChessTournamentManager** - Implementuje uživatelské rozhraní a využívá již vyjmenovaných projektů k provádění všech operací. Spojuje ostatní projekty do jedné aplikace.



Obrázek 4.1 Závislosti projektů

4.2.1 TournamentLibrary

Projekt *TournamentLibrary* je implementován jako knihovna poskytující základní operace související s párováním účastníků a generováním finálních výsledků, které mohou být dále využity v jiných aplikacích.

Každý typ turnaje je spravován jiným způsobem. Každému typu tedy odpovídá vlastní třída, kde jsou odlišnosti definovány. V projektu nalezneme třídy `RoundRobinTournament`, `SwissTournament`, `PlayoffTournament`, které dle názvu odpovídají danému turnaji. Jelikož některé činnosti jsou pro všechny turnaje stejné, je implementována rodičovská třída `TournamentHandler`, od které dědí třídy spravující konkrétní turnaje. Všechny třídy spravující turnaje navíc implementují interface `ITournament`, abychom v programu mohli turnaj spravovat nezávisle na jeho typu.

Dále si popíšeme jednotlivé entity, které se během správy turnaje vyskytují.

- `IParticipant` - Jakákoliv entita, která může hrát turnaj (hráč, tým ...). Objekty tříd implementující tento interface mohou být poté využity jako účastníci turnajů.

- `TournamentPlayer` - základní třída pro hráče turnaje
- `RoundDraw` - Reprezentuje rozlosování kola turnaje. Nalezneme zde jednotlivé zápasy.
- `IRoundPair` - zápas mezi entitami v turnaji
- `RoundPair` - konkrétní implementace `IRoundPair`
- `IMatchResult` - výsledek zápasu
- `SingleMatchResult` - Konkrétní implementace `IMatchResult` pro šachové turnaje, kde je možné ze zápasu získat žádný bod, půl bodu, nebo celý jeden bod
- `ParticipantWithPoints` - Obsahuje entitu hrající turnaj a zároveň body — včetně pomocných — získané v turnaji
- `ResultsSettings` - Počítání výsledků může obsahovat vlastní nastavení pro body za výhru a remízu. Nastavení je reprezentováno touto třídou.
- Extension metody - rozšíření především pro kolekce obsahující objekty výše uvedených tříd

Detaily o vlastnostech a metodách uvedených tříd nalezneme přímo v souboru s konkrétní třídou.

4.2.2 Tests

V projektu *Tests* nalezneme testy pro všechny projekty aplikace. Nachází se zde především testy na párování a generování výsledků turnajů. Projekt ale obsahuje i testy pro ostatní části aplikace. Pro testování se používají testovací frameworky *xUnit* a *bUnit*.

4.2.3 DatabaseCommunicator

Projekt *DatabaseCommunicator* reprezentuje vrstvu *Model*. Obsahuje především metody, které provádí dotazy nad databází a případně provádějí mapping výsledků dotazů na objekty.

Data jsou ukládána do databázového systému *Neo4j*. Abychom mohli s tímto systémem komunikovat, provádět dotazy nad daty a zpracovávat výsledky dotazů, využíváme oficiální knihovnu *Neo4j.Driver*. Oficiální knihovna dostala přednost před komerční *Neo4j.Client* z důvodu její kvalitní dokumentace a stability. Druhá zmíněná knihovna sice poskytuje automatický mapping komplexních výsledků dotazů, ale během testování nespolehlivě prováděla dotazy a mapování nefungovalo ve všech případech.

Jelikož v databázovém systému *Neo4j* nenalezneme fixně dané schéma jako například u relačních databází, je potřeba si definovat, jak jednotlivé údaje reprezentovat. Způsob reprezentace dat se odvíjí od UML diagramu z předchozí kapitoly. Nejprve si představíme jaké vrcholy se v databázi nacházejí, poté budeme

pokračovat hranami, kterými jsou propojeny. Nakonec si pro názornost předvedeme příklad konkrétní reprezentace.

Atributy jednotlivých vrcholů odpovídají třídám, které se nacházejí v projektu *DatabaseCommunicator* ve složce *Models*. S výjimkou vrcholů `RoundPair` a `ResultsSettings`, které využívají třídy přímo z projektu *TournamentLibrary*.

- `Address` - Odpovídá jakékoliv adrese. V současnosti se využívá pro určení lokace turnaje.
- `Player` - Reprezentuje každou osobu v aplikaci. Pojmenování `Player` bylo vybráno, jelikož i organizátoři hrají šachové turnaje a tudíž je toto pojmenování více reprezentativnější.
- `RegistrationSettings` - Reprezentuje nastavení registračních pravidel pro turnaj.
- `ResultsSettings` - Reprezentuje nastavení způsobu počítání výsledků.
- `Round` - Odpovídá kolu turnaje. Nemá vlastní třídu, jelikož v programu pro ní není využití.
- `RoundPair` - Reprezentuje zápas mezi hráči či týmy.
- `SwissTournamentSettings` - Reprezentuje nastavení švýcarského systému turnaje.
- `Team` - Reprezentuje tým složený z hráčů.
- `TeamDrawSettings` - Reprezentuje nastavení párování pro týmy.
- `Tournament` - Reprezentuje jeden určitý turnaj.
- `TimeControlSettings` - Reprezentuje nastavení časové kontroly turnaje.
- `TimeControlSettingsPiece` - Reprezentuje jednu sekce nastavení časové kontroly turnaje

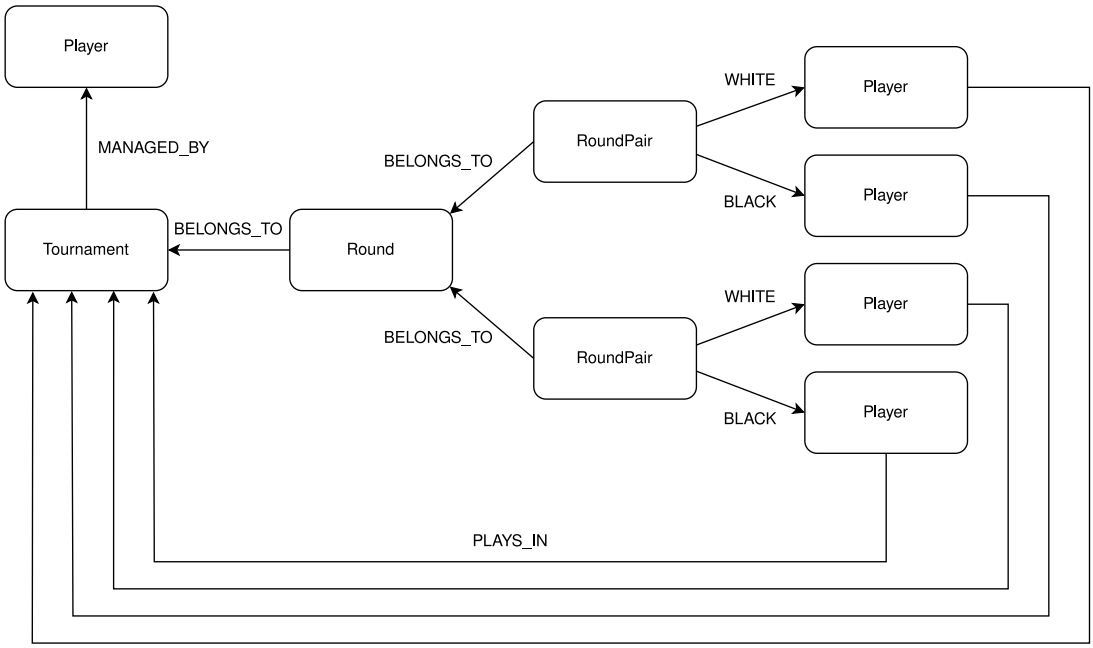
Následující seznam obsahuje hrany, které propojují jednotlivé vrcholy. U každé hrany je uveden krátký popis a všechny možnosti použití. Na levé straně šipky jsou možné počáteční vrcholy a na pravé strany jsou možné konečné vrcholy.

- `MANAGED_BY`
  - Určuje správce turnaje či týmu.
  - `Team, Tournament` → `Player`
- `PLAYS_FOR`
  - Hráč je členem týmu.
  - `Player` → `Team`

- BELONGS_TO
  - Vyjadřuje podřízenost k jiné entitě.
  - RoundPair → Round
  - Round → Tournament
  - TimeControlSettingsPiece → TimeControlSettings
- WHITE
  - Hráč nebo tým hraje v zápase jako bílý.
  - RoundPair → Player, Team
- BLACK
  - Hráč nebo tým hraje v zápase jako černý.
  - RoundPair → Player, Team
- LIKES
  - Turnaj byl uživatelem označen jako oblíbený.
  - Player → Tournament
- HAS_ADDRESS
  - Turnaj se hraje v lokaci.
  - Tournament → Address
- HAS_SWISS_SETTINGS
  - Nastavení švýcarského systému odpovídá.
  - Tournament → SwissTournamentSettings
- HAS_REGISTRATION_SETTINGS
  - Nastavení období registrace turnaje a maximální počet hráčů odpovídá.
  - Tournament → TournamentRegistrationSettings
- HAS_RESULTS_SETTINGS
  - Nastavení způsobu počítání výsledků turnaje odpovídá.
  - Tournament → ResultsSettings
- HAS_TEAMDRAW_SETTINGS
  - Nastavení turnaje pro týmy odpovídá.
  - Tournament → TeamDrawSettings
- HAS_TIME_SETTINGS
  - Nastavení časové kontroly turnaje odpovídá.
  - Tournament → TimeControlSettings

- `SUBPAIR`
  - Zápas mezi hráči patří k zápasu týmu.
  - `RoundPair` → `RoundPair`

Pro lepší znázornění si předvedeme základní příklad, což je turnaj jednotlivců s jedním vygenerovaným kolem. Turnaj je reprezentován jedním vrcholem typu `Tournament`. Pro určení správce turnaje je použita hrana `MANAGED_BY`, která spojuje turnaj a uživatele (vrchol typu `Player`) spravujícího daný turnaj. Hráči, kteří jsou registrovaní na turnaj, jsou reprezentováni vrcholem typu `Player` a jsou spojeni s turnajem hranou `PLAYS_IN`. Vygenerované kolo, reprezentované vrcholem typu `Round`, je opět připojeno k vrcholu znázorňující turnaj hranou `BELONGS_TO`. Zápasy, které se odehrávají v daném kole jsou reprezentovány vrcholy typu `RoundPair` a připojeny ke kolu turnaje hranou `BELONGS_TO`. Nakonec jsou hráči přiřazeni do určitého zápasu jako černí nebo bílí, což je reprezentováno hranami typu `WHITE`, respektive `BLACK`.



Obrázek 4.2 Ukázka turnaje v databázi

Pro provedení dotazu nad popsaných schématem využijeme tříd, které dědí od `DatabaseModelManipulator`. Rodičovská třída implementuje základní metody pro přístup k databázi. Pro spuštění dotazu vždy použijeme metodu `AsyncSession` nad zděděným objektem `_driver`, který zajišťuje přímý přístup do databáze. Nad získanou session poté provedeme dotaz. Abychom zabránili potencionálnímu útoku, parametry do dotazu jsou vkládány vždy metodou `RunAsync` nad danou session.

```
public class DemoManipulator : DatabaseModelManipulator
{
    public async Task PerformQuery()
    {
        var parameters = new
        {
            DemoProperty = DemoValue
        };
        using (var session = _driver.AsyncSession())
        {
            string query = "DEMO $DemoProperty QUERY";
            IResultCursor result = await
                session.RunAsync(query, parameters);
        }
    }
}
```

4.2.4 ChessTournamentManager.Client

Projekt *ChessTournamentManager.Client* slouží jako projekt, kde mohou být implementovány komponenty, které běží v prohlížeči uživatele. Jelikož v našem projektu tuto funkcionalitu nevyužíváme, nachází se zde pouze základní části kódu. Projekt zde zůstává pro případné budoucí využití.

4.2.5 ChessTournamentManager.Shared

Do projektu *ChessTournamentManager.Shared* jsou implementovány různé kusy kódu, které provádí základní činnosti a využijeme je v ostatních projektech. Nachází se zde různé extension metody, třídy obsahující možné typy jednotlivých turnajů aj.

Jako nejdůležitější konstrukce z projektu můžeme uvést třídy ve složce *QuickResponseMessages*. Základní třídou je `QuickResponseMessage`, která reprezentuje jakoukoliv zprávu, kterou bychom chtěli uživateli zobrazit v uživatelském rozhraní. Její vlastnost `MessageType` určuje typ zprávy, pomocí které je poté rozhodnuto o barvě zprávy v uživatelském rozhraní a dalších činnostech. Ostatní třídy, `SuccessfulMessage`, `UnsuccessfulMessage`, `NotExistsInDatabase`, dědí od zmíněné třídy `QuickResponseMessage` a mají již přednastavený `MessageType` a tudíž usnadňují používání zpráv a zvyšují čitelnost kódu.

4.2.6 ChessTournamentManager

Jádrem naší aplikace je projekt *ChessTournamentManager*, který využívá již představené projekty a propojuje je do funkčního celku za pomoci frameworku Blazor.

Základním stavebním kamenem jsou komponenty nacházející se v adresáři *Components*. Každá komponenta se musí nacházet přímo v tomto adresáři či jeho podadresářích. Kompilátor ji jinak nebude považovat za platnou. Dále si popíšeme již existující podadresáře v adresáři *Components*.

- **Layout** - Tento adresář tvoří základ uživatelského rozhraní aplikace. V souboru *MainLayout.razor* je implementováno rozložení aplikace — menu aplikace, horní lišta a zbývající prostor určený pro vykreslování obsahu konkrétní stránky dle URL. Druhý soubor *MainPanel.razor* představuje hlavní menu aplikace a slouží k jednoduchému zobrazování dalších stránek.
- **Account** - Zde se nachází všechny komponenty a stránky související se správou účtů uživatelů.
- **Pages** - Obsahuje všechny stránky v aplikaci. Odpovídá tedy všem komponentám, které jsou označeny atributem `@page`.
- **TournamentTabs** - Zde se nachází všechny komponenty, které implementují záložky správy turnaje nebo s nimi úzce souvisí.

Většina dat je ukládána do databáze. Výjimkami jsou pouze profilové obrázky pro účty i turnaje, které jsou ukládány jako soubory. V adresáři *ProfilePictures*, se nacházejí defaultní obrázky, které se použijí, pokud uživatel nenahraje vlastní.

4.3 Lokalizace

Abychom mohli poskytnout uživatelům možnost volby jazyka, je potřeba psát kód, který bude podporovat lokalizování textů viditelných v uživatelském rozhraní. Abychom se odprostili od technických detailů, využíváme knihovnu `Microsoft.Extensions.Localization` od společnosti Microsoft, která poskytuje třídy a funkce pro překlad textových řetězců do vybraného jazyka. Výběr jazyka obstarává komponenta `CultureSelector`, která poskytuje uživateli možnost výběru jazyka v uživatelském rozhraní. Abychom uživatele nezatěžovali nutností vždy vybírat preferovaný jazyk, ukládáme vybranou hodnotu jako Cookie v prohlížeči.

Překlady do jednotlivých jazyků jsou uloženy v adresáři `LanguageAssets` v projektu `ChessTournamentManager`. Každý podadresář obsahuje překlady vybrané části aplikace. Soubor s defaultními hodnotami je vždy pojmenován jako `Resource.resx`. Překlady do dalších jazyků jsou poté uloženy ve stejném adresáři a pojmenovány jako `Resource.KODJAZYKA.resx`, kde `KODJAZYKA` odpovídá vybranému kódu jazyka dle standardu ISO 639-1. Pokud poté potřebujeme provést lokalizaci textu, použijeme implementaci rozhraní `IStringLocalizer<Resource>`, kterou nám poskytne dependency injection kontejner. Získaný objekt vrátí překlad dle poskytnutého klíče.

4.4 Jmenné konvence

Při vývoji každého projektu, musíme dbát na styl pojmenovávání. Při psaní kódu dodržujeme základní konvence, které se odvíjí od doporučení společnosti Microsoft pro psaní kódu v jazyce C# [13]

- Názvy tříd jsou psané stylem *PascalCase*.
- Lokální proměnné jsou psané stylem *camelCase*.

- Private a protected pole jsou psané stylem *camelCase* a název vždy začíná podtržítkem.
- Public vlastnosti jsou psané stylem *PascalCase*.
- Názvy interfaců jsou psané stylem *PascalCase* a název začíná velkým měkkým I.

4.5 Autentifikace a authorizace

4.5.1 Správa uživatelských účtů

Aplikace využívá předvytvořenou autenfitikaci i authorizaci, které jsou vygenerovány při založení projektu *Blazor WebApplication* v programu Visual Studio 2022[1]. Dojde k vygenerování kódu na manipulaci s uživateli, uživatelského rozhraní pro správu účtu i databázových struktur na ukládání dat o účtech v databázovém systému *Microsoft SQL Server*. Pro přihlášení je nutné znát e-mail a heslo, pomocí nichž dojde k autentifikaci uživatele.

Účtu jsou ukládány v samostatné databázi odděleně od informací o turnajích a hráčích z důvodu bezpečnosti.

4.5.2 Využití účtů v datové databázi

Každý uživatel, který chce aktivně pracovat s programem, musí mít vytvořený účet. Jak již bylo zmíněno, informace o účtu jsou uloženy v samostatné databázi oddělené od ostatních dat. Každého uživatele reprezentuje v datové databázi jeden vrchol typu `Player`. Do vlastnosti `AccountId` přiřadíme hodnotu ze sloupce `Id` z tabulky `AspNetUsers` odpovídajícího uživatele. Tímto způsobem provážeme vrchol `Player` s uživatelským účtem. Pracovat přímo s databází s účty by přinášelo bezpečnostní riziko. Hodnotu tedy získáme pomocí metody `GetLoggedUserIdAsync` implementované ve třídě `UserInformation`. Metoda využívá metody poskytnuté frameworkem Blazor pro bezpečné a efektivní získání tohoto identifikátoru.

4.5.3 Konkrétní využití autentifikace a autorizace

Některé části aplikace jsou dostupné pro všechny uživatele. Naopak některé části vyžadují přihlášení. Stránkám, které jsou dostupné pro všechny uživatele, nemusíme věnovat žádnou zvláštní pozornost. Naopak stránky, které jsou dostupné pouze přihlášeným uživatelům, označíme v kódu atributem `@attribute [Authorize]`. V případě, že uživatel během své činnosti narazí na stránku určenou pouze pro přihlášené uživatele a není ještě přihlášen, dojde k automatickému přesměrování na přihlašovací stránku. Po úspěšném přihlášení bude moci uživatel dále pokračovat v činnosti. Tímto označením, ale stále není zabráněno uživatelům provádět činnosti, které jsou omezeny pouze na konkrétní uživatele. Například turnaj může spravovat pouze jeho správce. Zde využijeme vlastnosti `AccountId` vrcholů typu `Player`. Před provedení činnosti, kterou uživatel vyžaduje, vždy

[1] learn.microsoft.com/en-us/aspnet/core/blazor/security/?view=aspnetcore-8.0

ověříme, zda je uživatel oprávněn k této akci. Opět zjistíme `Id` aktuálně přihlášeného uživatele pomocí již zmíněné metody `GetLoggedUserIdAsync` a v závislosti na konkrétní situaci rozhodneme, zda je uživatel oprávněn danou akci provést.

Nakonec ještě zmíníme jednu užitečnou vlastnost frameworku Blazor. V případech, kdy potřebujeme zobrazit rozdílný obsah stránky pro přihlášené a nepřihlášené uživatele, využíváme tagů `<AuthorizedView>` a `<NotAuthorizedView>`. Obsah uvnitř těchto tagů je buď skryt či zobrazen uživateli na základě jeho stavu přihlášení.

4.6 Přesměrování na jinou stránku v aplikaci

Každá stránka v aplikaci má svůj URL. Pokud bychom chtěli zobrazit stránku na jiném URL, běžně využíváme implementaci třídy `NavigationManager` z Blazoru, kterou získáme z dependency injection kontejneru. Objektu poté vždy poskytneme URL v podobě textového řetězce a objekt provede přesměrování. Tímto způsobem ale mohou vznikat chyby způsobené nepozorností při psaní URL. Navíc pokud by došlo ke změně URL stránky, bylo by nutné změnit konkrétní řetězce pro přesměrování na jiných stránkách. V aplikaci je tedy implementována třída `NavigationUrls`, která tyto problémy eliminuje. Obsahuje metody, které provádějí přesměrování na požadovanou stránku. Pokud by došlo k přesunu obsahu na jiné URL, stačí pouze změnit obsah dané metody a přesměrování z jiných stránek bude stále funkční, jelikož název metody zůstane neměnný. Též nevznikají chyby kvůli překlepům v odkazech – vše je kontrolováno kompilátorem.

4.7 Frontend

Framework Blazor nám umožňuje upravovat vzhled elementů pomocí CSS či jeho rozšíření SASS[2] nebo LESS[3]. To by ale znamenalo nutnost úpravy jednotlivých elementů v aplikaci zaměřené více na backend. Využíváme tedy frontend frameworku Bootstrap[4]. Jelikož se u naší aplikace očekává užívání na různých typech zařízení s odlišnou velikostí displejů, je nutné, aby bylo rozhraní aplikace responzivní. *Bootstrap* poskytuje možnosti jak této funkcionality efektivně dosáhnout. Aplikace využívá jeho *Layout* funkcí, o kterých se lze více dozvědět na oficiální stránce[5].

Jelikož funkce, které poskytuje *Bootstrap*, nelze modifikovat pomocí proměnných v jazyce C#, využíváme též jeho rozšíření pro framework Blazor v podobě knihovny *BlazorBootstrap*[6]. Knihovna využívá výhod *Bootstrapu* i výhod frameworku Blazor najednou. Poskytuje nám již předpřipravené komponenty, které je možné dynamicky modifikovat pomocí parametrů a funkcí.

V případě, že na dosažení požadovaného vzhledu nedostačují knihovny *BlazorBootstrap* ani základní *Bootstrap* je využito CSS. CSS pravidla, která jsou

[2] sass-lang.com/documentation
[3] lesscss.org
[4] getbootstrap.com
[5] getbootstrap.com/docs/5.0/forms/layout
[6] docs.blazorbootstrap.com

dostupná globálně pro všechny komponenty, se nacházejí v projektu *ChessTournamentManager* v adresáři *wwwroot* v souboru *app.css*. Pravidla není nutná do komponent nijak importovat, ale lze je okamžitě používat. Stačí pouze využít daná pravidla u konkrétních tagů. Pokud bychom potřebovali pravidlo pouze pro vybranou komponentu, je možné vytvořit soubor, který se bude nacházet na stejné cestě jako soubor s komponentou, pro kterou je určený. Název souboru s CSS pravidly musí být pojmenován totožně jako soubor s komponentou. Následně je k názvu tohoto souboru přidána koncovka *.css*. Například pokud je název souboru s komponentou je `XXX.razor`, tak název souboru s CSS pravidly musí být `XXX.razor.css`.

Na závěr části o frontendu aplikace si popíšeme práci s vlajkami zemí. Na stránce s profily uživatelů totiž zobrazujeme vlajku státu, který reprezentují. Stáhnout vlajky všech zemí a vhodně je naformátovat by nám zabralo určité množství času. Z tohoto důvodu aplikace využívá knihovnu *Blazor.Flags*[7], která poskytuje základní třídy pro práci se státy a jejich vlajkami. Tyto třídy jsou využívány pro zobrazení vlajek země právě na profilu hráče, či ve formulářích s názvy států.

4.8 Nasazení a spuštění programu

Aby bylo nasazení programu v produkčním prostředí co možná nejjednodušší, využíváme programu *Docker*. Ten je nutné si nejprve nainstalovat. V souboru *Dockerfile* jsou příkazy pro *Docker*, které provedou zkompilování samotné aplikace bez dalších závislostí. Jelikož pro ukládání dat jsou použity databázové systémy *Neo4j* a *Microsoft SQL Server 2019*, je součástí zdrojových souborů i soubor *docker-compose.yml*, ve kterém nalezneme definice čtyř kontejnerů:

- **neo4j** - kontejner poskytující databázový systém *Neo4j*; využívá oficiální image pro tento databázový systém[8]
- **sqlserver** - kontejner poskytující databázový systém *Microsoft SQL Server 2019*; využívá oficiální image pro tento databázový systém[9]
- **crateaccountdatabase** - kontejner, který vytvoří databázi pro ukládání účtů v kontejneru *sqlserver*
- **chesstournamentmanagerweb** - kontejner samotné aplikace

Program tedy můžeme spustit ve složce, kde se nachází *Dockerfile*, příkazem

```
docker-compose up --build
```

čímž dojde ke stažení potřebných závislostí, zkompilování aplikace i jejímu následnému spuštění. Aplikace běží na portu **9000**. Přístup pro běžné uživatele nastavíme dle konkrétního nasazení.

[7] github.com/kevinvenclovas/Blazor.Flags
[8] hub.docker.com/_/neo4j
[9] hub.docker.com/_/microsoft-mssql-server

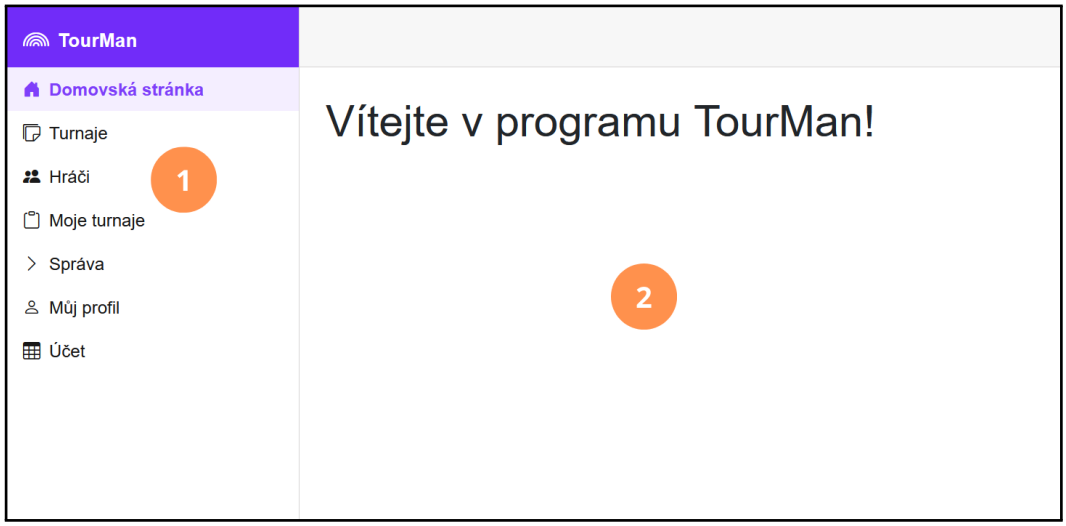
Při nasazení aplikace také musíme vyřešit otázku zasílání e-mailových zpráv, které jsou využívány pro potvrzování registračních e-mailů. V souboru *appsettings.json* je nutné nastavit e-mailovou schránku odkud budou e-mailové zprávy odesílány. V případě nenastavení této schránky nebude docházet k odesílání potvrzovacích e-mailových zpráv a uživatelé se nebudou moci registrovat.

5 Uživatelská dokumentace

V uživatelské sekci si nejprve představíme jednotlivé záložky a jejich funkce, abychom získali lepší přehled o struktuře aplikace. Poté si ukážeme, jak provést vybrané úkony jako organizátor i jako hráč.

Popisy uživatelského rozhraní jsou pro větší názornost doplněny obrázky zobrazující popisovanou činnost. V závorce za popisem akce se nachází číslo, které je též na obrázku, sloužící k identifikace elementu, který je právě popisován.

Na levé straně aplikace se nachází hlavní panel (1), který slouží jako menu pro výběr skupiny činností. Po kliknutí na danou položku se zobrazí příslušný obsah ve středové části aplikace (2), kde lze provádět konkrétní akce. V dalších sekcích si popíšeme, co příslušné záložky obsahují a jaké činnosti na nich můžeme dělat.



Obrázek 5.1 Rozložení aplikace

5.1 Základní záložky

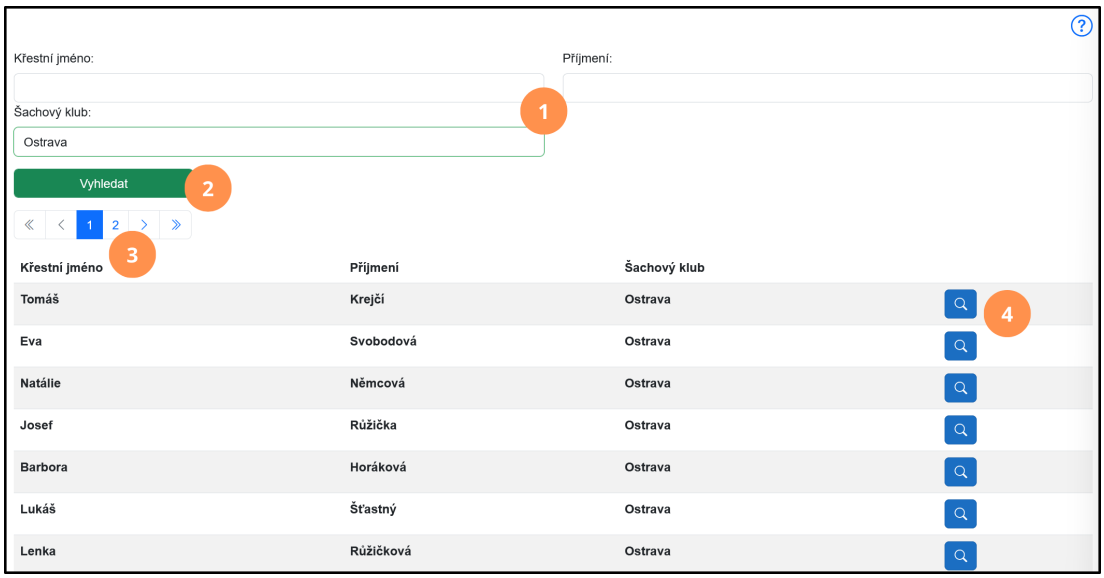
Nejprve se podíváme na záložky, které jsou využívány většinou uživatelů aplikace.

5.1.1 Záložky Registrace a Přihlášení

Aplikaci je možné používat i bez vytvoření účtu. To nás ale omezuje pouze na pasivní prohlížení dat bez možnosti provádění pokročilých akcí. Na záložce *Registrace*, si můžeme vytvořit nový účet. Zadáme naši e-mailovou adresu i heslo, a poté potvrdíme naši e-mailovou adresu kliknutím na odkaz, který přijde do e-mailové schránky. Tím dojde k potvrzení registrace. Při dalším používání aplikace se již stačí pouze přihlásit na záložce *Přihlášení*.

5.1.2 Záložka Hráči

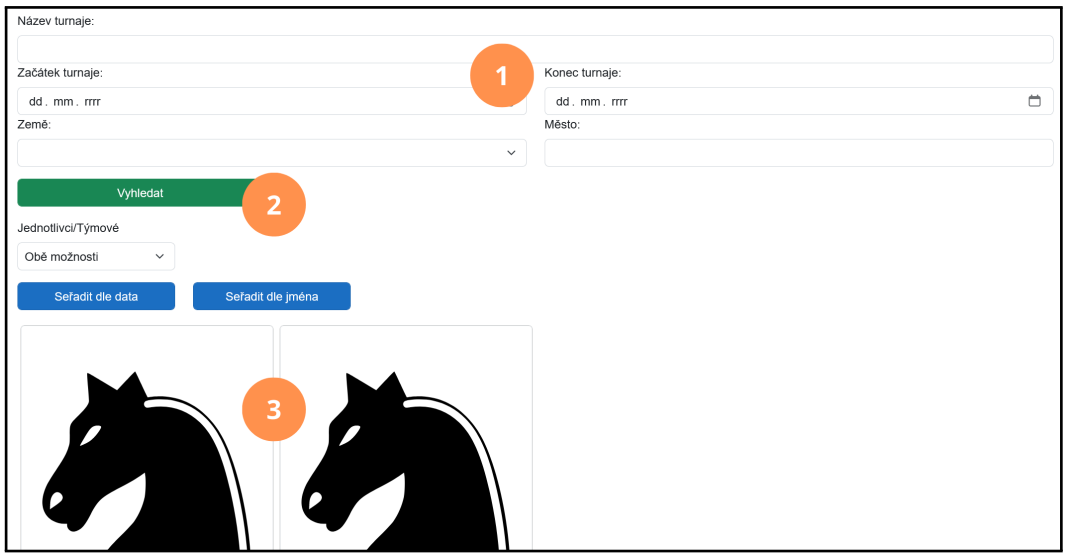
Na záložce *Hráči* se po vyplnění jednotlivých údajů (1) a následném kliknutí na tlačítko *Vyhledat* (2) zobrazí pouze hráči, kteří odpovídají vyplněných údajům. Položku je též možné nevyplnit a tím pádem nebude ve vyhledávání zohledněna. Pokud je hráčů splňující daná kritéria více než deset, zobrazí se jich na dané stránce pouze část. Další hráče je možné zobrazit přepnutím na další stránku (3). U každého hráče se nachází tlačítko s ikonou lupy (4), jehož kliknutím dojde k zobrazení profilu daného hráče, kde jsou doplňující osobní informace.



Obrázek 5.2 Záložka Hráči

5.1.3 Záložka Turnaje

V případě, že bychom se potřebovali registrovat na turnaj či zobrazit jeho výsledky, zobrazíme si záložku *Turnaje*. Vyplněním hodnot ve formuláři (1) a kliknutím na tlačítko *Vyhledat* (2) se zobrazí pouze turnaje odpovídající daným podmínkám (3). U každého turnaje je tlačítko *Zobrazit detaily*, které zobrazí stránku se všemi informacemi o turnaji, včetně výsledků, rozlosování a dalších. Též se u každého turnaje nachází tlačítko *Registrovat se*, které zobrazí stránku, kde je možné se registrovat na daný turnaj.



Obrázek 5.3 Záložka Turnaje

5.1.4 Záložka Moje turnaje

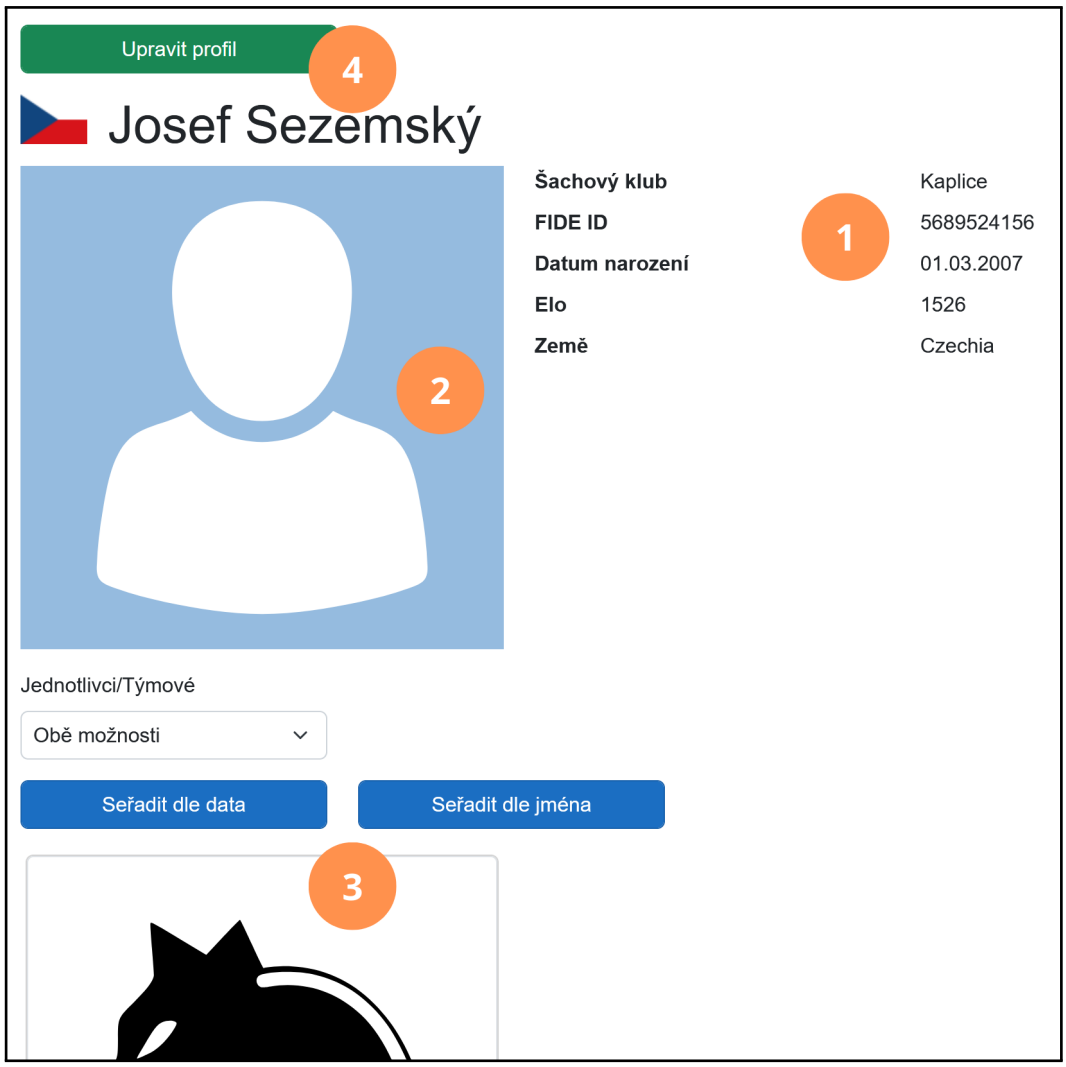
Záložka *Moje turnaje* zobrazuje všechny turnaje, ve kterých jsme vedeni jako účastník nebo které jsme označili jako oblíbené. Turnaje jsou rozděleny do tří sekcí:

- **Současné a nadcházející turnaje** - Turnaje, ve kterých jsme registrováni jako hráč nebo jako člen týmu. Jsou zde zobrazeny pouze turnaje, které v současnosti probíhají a nebo proběhnou v budoucnu.
- **Oblíbené turnaje** - Turnaje, které jsme označili jako oblíbené. Lze využít ke sledování turnajů svých oblíbených hráčů či spoluhráčů z týmu.
- **Proběhlé** - Turnaje, ve kterých jsme registrováni jako hráč nebo jako člen týmu, ale turnaje již byly ukončeny.

V sekci *Současné a nadcházející turnaje* je možné po kliknutí na tlačítko *Zrušit registraci* u vybraného turnaje zobrazit stránku, kde je možné provést odhlášení z turnaje.

5.1.5 Záložka Můj profil

Svůj osobní profil můžeme vidět na záložce *Můj profil*, kde jsou zobrazeny naše osobní údaje (1) společně s profilovou fotkou (2) a zároveň turnaje, kterých jsme se účastnili nebo se teprve budeme účastnit (3). Změnu osobních informací či nahrání nové profilové fotky lze provést na stránce, která se zobrazí po kliknutí na tlačítko *Upravit profil* (4).



Obrázek 5.4 Záložka Můj profil

V zobrazeném formuláři je možné vyplnit osobní údaje (1), které se uloží po kliknutí na tlačítko *Uložit* (2). Též je možné nahrát zmíněný profilový obrázek, který bude zobrazen na našem profilu (3). Nahrání obrázku je opět nutné potvrdit příslušným tlačítkem s nápisem *Uložit* (4). Možnost nahrát profilový obrázek se zobrazí na stejné stránce pouze pokud jsou vyplněny a uloženy osobní informace. Zobrazit opět stránku s naším profilem lze kliknutím na tlačítko *Zpátky na profil*.

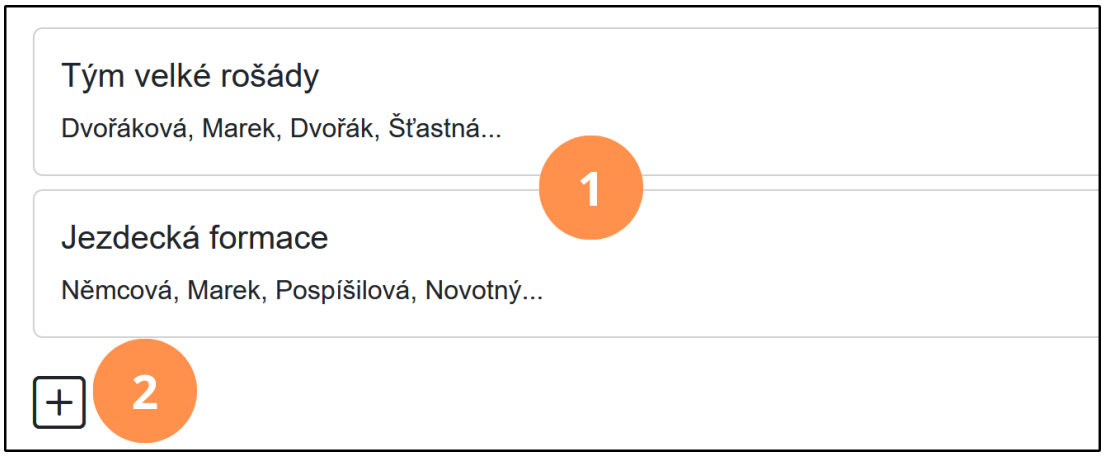
Obrázek 5.5 Záložka Můj profil - změna informací

5.2 Záložky pro pokročilé uživatele

Výše představené záložky jsou určeny především pro účastníky turnajů jednotlivců, kteří tvoří většinu uživatelů aplikace. Zbývající záložky, které jsou sbalené do společného kontejneru s pojmenováním *Správa* v hlavním panelu, slouží ke správě týmů a turnajů.

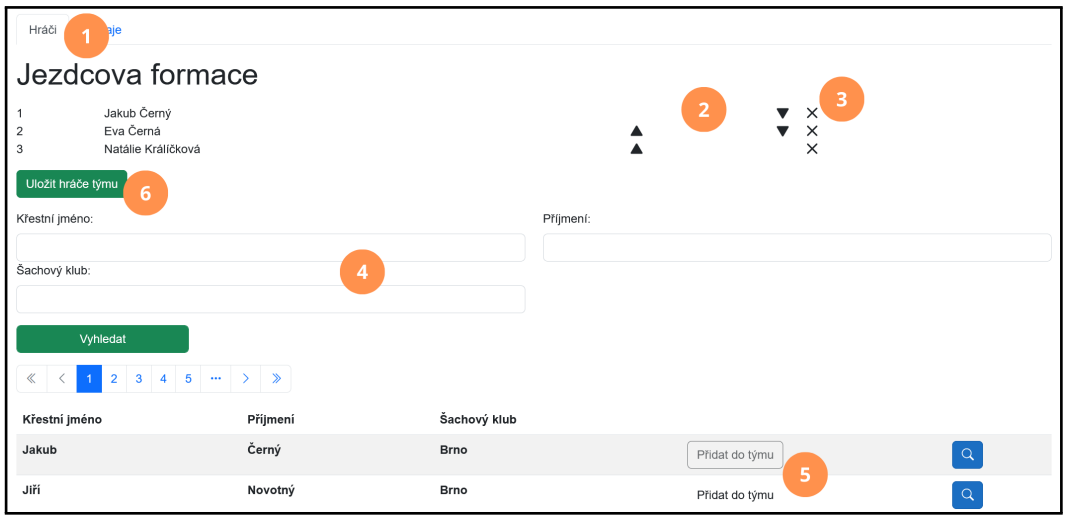
5.2.1 Záložka Spravované týmy

Na záložce *Spravované týmy* nalezneme všechny týmy, které spravujeme (1). Kliknutím na daný tým se zobrazí stránka, kde lze upravovat hráče týmu a zobrazit turnaje, na které je daný tým přihlášen, a případně z nich tým odhlásit. Vytvořit nový tým můžeme pomocí tlačítka se symbolem „+“ (2).



Obrázek 5.6 Záložka Spravované týmy - seznam týmů

Změnit hráče v týmu lze v sekci *Hráči* (1), nicméně je to možné pouze pokud tým již není přihlášený na turnaj. Pomocí šipek u jména hráče, můžeme měnit jeho pořadí v týmu (2). Křížkem naopak odstraníme hráče z týmu (3). Pokud bychom chtěli přidat nového hráče, vyhledáme ho dle formuláře (4) a klikneme na tlačítko *Přidat do týmu* (5). Všechny změny je nutné uložit tlačítkem *Uložit hráče týmu* (6).

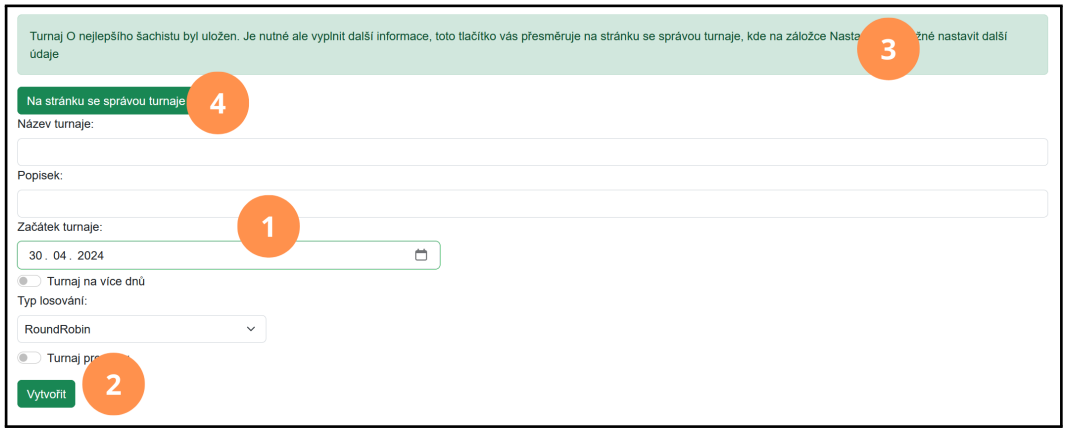


Obrázek 5.7 Správa týmu - hráči

V sekci *Turnaje* můžeme vidět všechny turnaje, na které je tým registrován. Turnaje jsou rozděleny stejně jako na záložce *Moje turnaje* v hlavním panelu. Kliknutím na tlačítko *Zrušit registraci* dojde k zobrazení stránky, kde je možné daný tým odhlásit z turnaje.

5.2.2 Záložka Vytvoření turnaje

Pokud bychom chtěli uspořádat turnaj, je nutné ho nejprve vytvořit. Na záložce *Vytvořit turnaj* se nachází formulář, jehož vyplněním (1) a odesláním (2) dojde k vytvoření nového turnaje. Při úspěšném vytvoření se zobrazí potvrzující zpráva (3) a tlačítko, jehož kliknutí zobrazí stránku, kde lze daný turnaj spravovat (4).



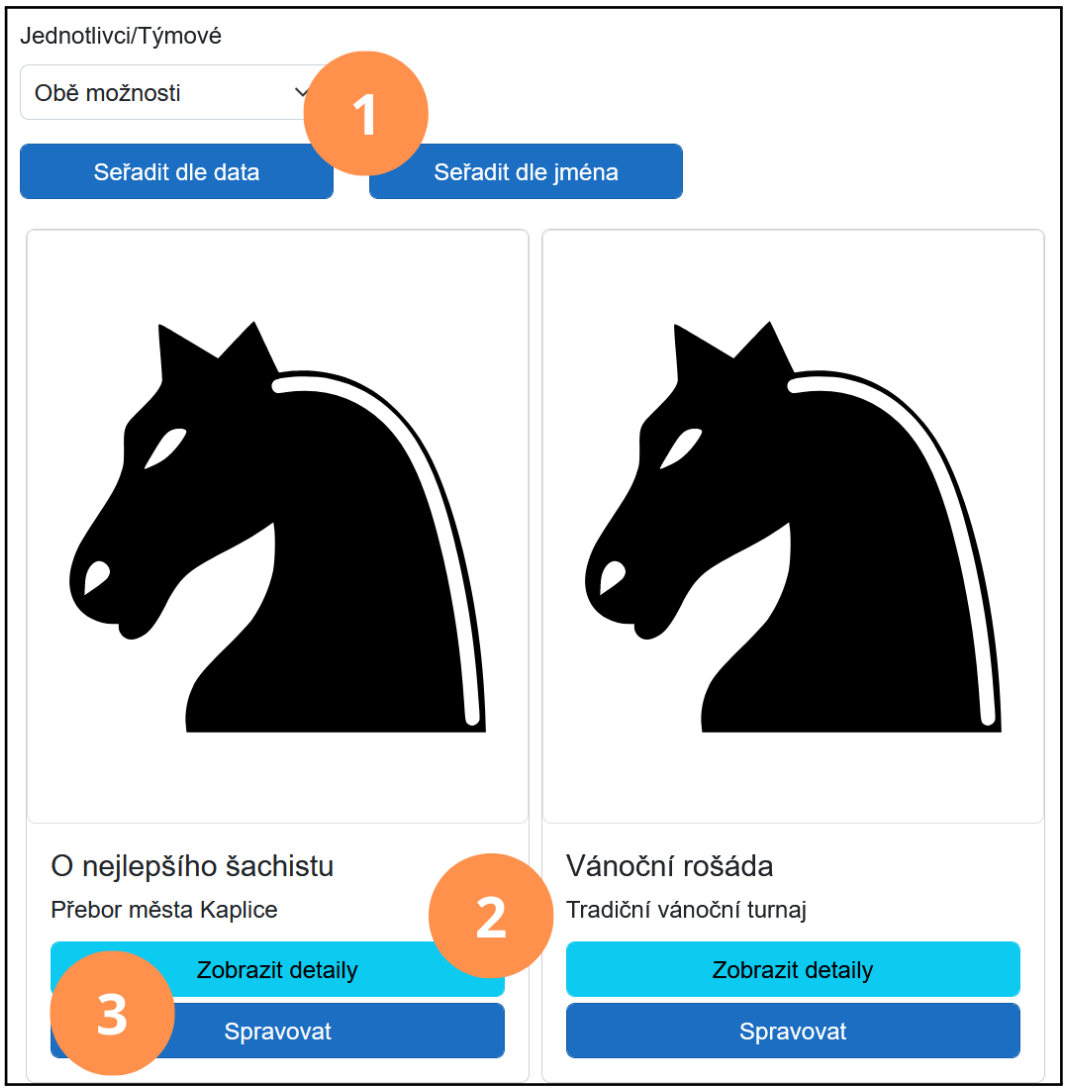
Obrázek 5.8 Záložka Vytvoření turnaje

Formulář na vytvoření turnaje obsahuje pouze základní nastavení, další nastavení lze provést v sekci *Nastavení* na stránce se správou turnaje (viz záložka

Spravované turnaje). Údaje vyplněné v tomto formuláři již nelze po vytvoření turnaje dále měnit. Toto omezení je z důvodu zabránění zmatení hráčů, kteří se již na turnaj přihlásili. Údaje ve formuláři představují základní údaje a například v případě změny termínu by již někteří soutěžící nemuseli se změnou souhlasit.

5.2.3 Záložka Spravované turnaje

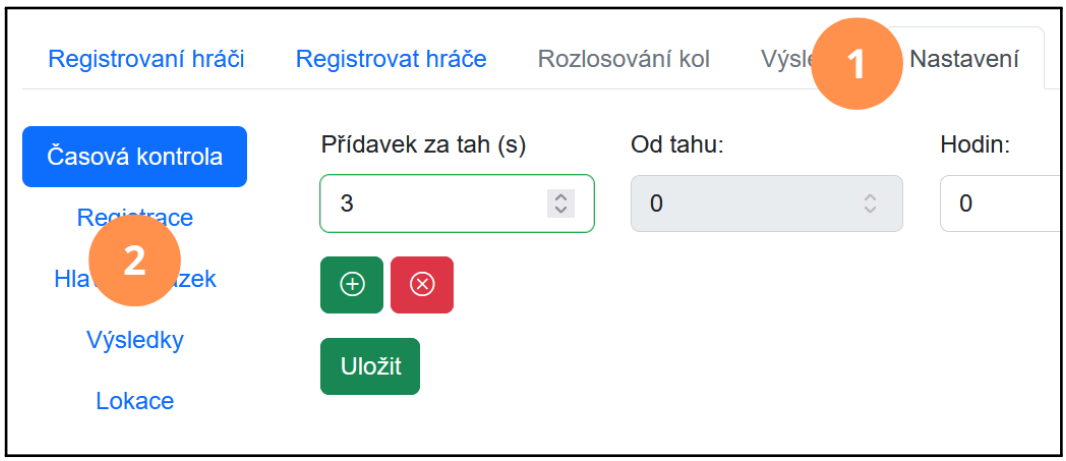
Pokud jsme již vytvořili nějaké turnaje a chtěli bychom je spravovat, nalezneme je na záložce *Spravované turnaje*. Turnaje můžeme filtrovat dle typu a řadit dle jména i data (1). Kliknutím na tlačítko *Zobrazit detaily* (2) se zobrazí stránka s informacemi o turnaji, jak ji uvidí potencionální účastníci. Kliknutím na tlačítko *Spravovat* (3) u konkrétního turnaje bude zobrazena stránka, kde je možné provádět veškeré akce, kterou souvisí se správou turnaje. Jednotlivé sekce si nyní představíme.



Obrázek 5.9 Záložka Spravované turnaje - seznam turnajů

Nastavení

Aby turnaj proběhl dle našich představ, musíme aplikaci poskytnout dodatečné informace o turnaji. Údaje vyplníme v sekci *Nastavení*, která obsahuje několik podsekcí. Vyplněné informace uvidí potencionální účastníci turnaje a dle nich se budou moci rozhodnout o účasti v turnaji.



Obrázek 5.10 Správa turnaje - rozložení nastavení

V podsekci *Časová kontrola* můžeme určit tempo šachových partií. Po kliknutí na zelené tlačítko se symbolem „+“ přidáme novou část, kterou lze případně odebrat červeným tlačítkem se symbolem „×“. Nastavení časové kontroly uložíme pomocí tlačítka *Uložit*. Zvolené údaje časové kontroly slouží k automatickému určení typu turnaje (*Blitz, Rapid, Classical*).

Abychom umožnili hráčům i týmům se přihlásit na turnaj, je nutné jim určit časové období, během kterého bude možné registrace provádět, což lze udělat v podsekci *Registrace*. Dle prostorových i časových možností zvolíme maximální počet hráčů nebo týmů, kterým bude umožněno se turnaje účastnit .

K upoutání hráčů či zobrazení loga turnaje nebo sponzora je možné v podsekci *Hlavní obrázek* nahrát hlavní obrázek. Ten bude poté viditelný na několika místech v aplikaci, kde jsou turnaje zobrazeny i s obrázky. Pokud není nahrán vlastní obrázek, bude zobrazen základní obrázek se šachovým koněm.

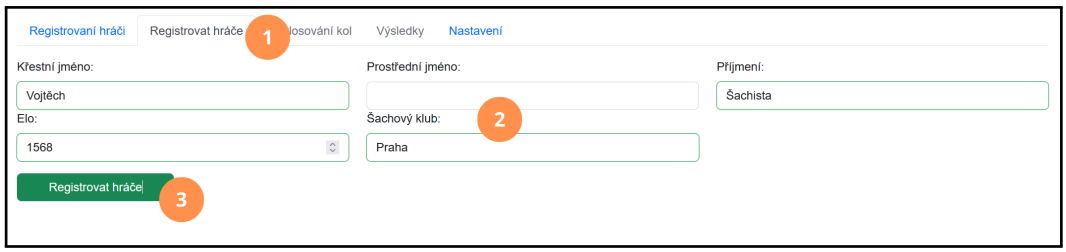
V případě týmového turnaje je důležité v podsekci *Nastavení týmů* nastavit omezení velikosti týmů dle počtu hráčů, které se mohou do turnaje přihlásit. Je důležité zvolit tuto hodnotu, jinak se nebudou moci na turnaj přihlásit žádné týmy, jelikož hodnota je v základu nastavena na mínus jedna.

V základním nastavení se celkový počet bodů určí dle bodů získaných z partií. V případě, že bychom chtěli definovat jiné bodové schéma, můžeme v podsekci *Výsledky* nastavit, kolik bodů je za vítězství v zápase a kolik za remízu.

Registrace hráčů

Po vytvoření turnaje je turnaj viditelný pro ostatní uživatele, kteří si mohou turnaj vyhledat a registrovat se na něj. Pokud by se ale v den turnaje objevil

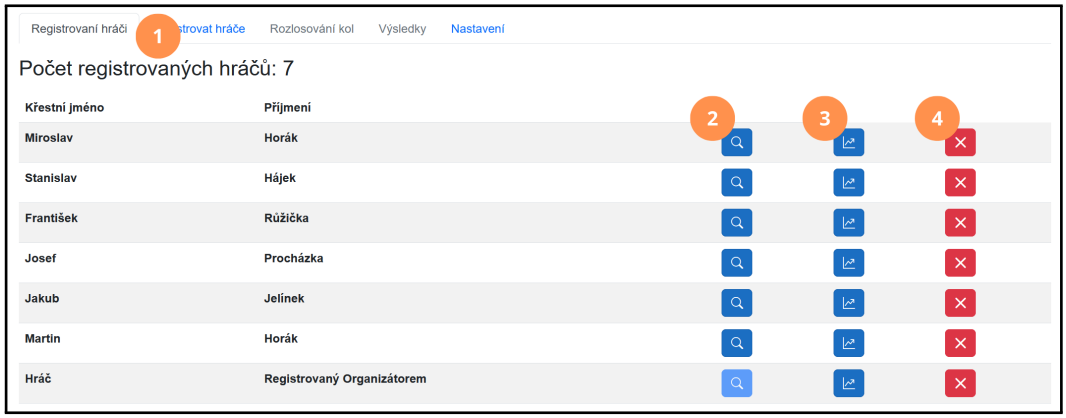
neznámý hráč, je možné jej před začátkem turnaje dodatečně registrovat, což provedeme v sekci *Registrovat hráče* (1). Vyplněním formuláře s údaji o hráči (2) a následným kliknutím na tlačítko *Registrovat hráče* (3), dojde k uložení údajů o hráči a jeho přihlášení na turnaj. Aby bylo možné hráče přihlásit, je nutné vyplnit alespoň křestní jméno a příjmení.



Obrázek 5.11 Správa turnaje - registrace neznámého hráče

Zobrazení registrací hráčů

Na záložce *Registrovaní hráči* (1) nalezneme všechny dosud registrované hráče na turnaji. U každého hráče nalezneme tlačítko se symbolem lupy (2), po jehož kliknutí dojde k přesměrování na profil hráče. V případě, že se jedná o hráče, kterého jsme manuálně registrovali, není možné na tlačítko kliknout z důvodu neexistujícího profilu. Dále zde nalezneme tlačítko, které nás přesměruje na stránku se zápasy a výsledky hráče v turnaji (3). Jako poslední zde nalezneme červené tlačítko se symbolem křížku (4), pomocí něhož můžeme odhlásit hráče z turnaje. Nesmíme opomenout, že tlačítko na odhlášení není viditelné poté, co turnaj již začal. Dostupné není ani v případě týmového turnaje, ze kterého je možné odhlásit pouze celé týmy, nikoliv hráče jako jednotlivce.



Obrázek 5.12 Správa turnaje - registrovaní hráči

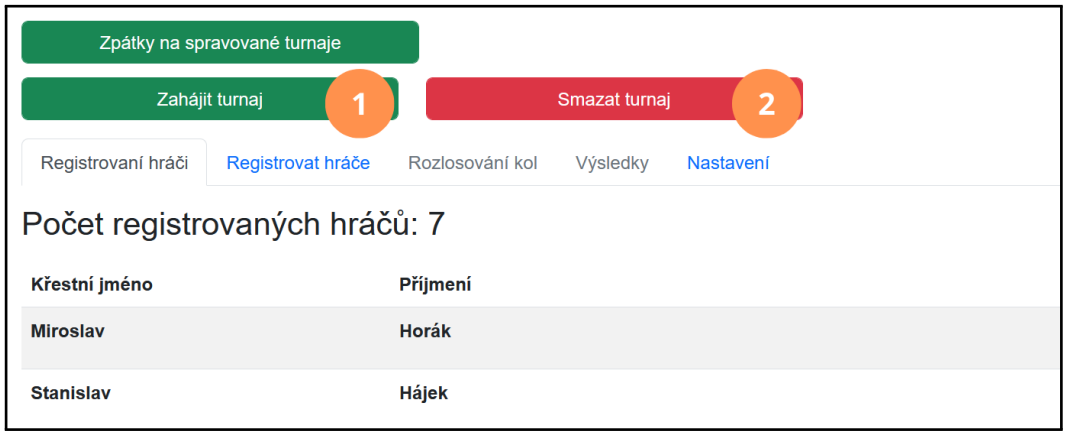
Zobrazení registrací týmů

V případě, že turnaj je určen pro týmy, nalezneme na stránce se správou turnaje i sekci *Registrované týmy*, kde jsou zobrazeny všechny registrované týmy. Po kliknutí na daný tým dojde k zobrazení hráčů týmu, u kterých opět můžeme zobrazit jejich profil a zápasy v turnaji. Týmy můžeme odhlásit z turnaje pomocí červeného tlačítka se symbolem „×“. Odebrání týmu z turnaje opět není možné po jeho zahájení.

Smazaní a zahájení turnaje

Zahájit turnaj je možné tlačítkem *Zahájit turnaj* (1), čímž dojde k zablokování příchozích registrací, možnosti registrovat nové hráče a možnosti měnit nastavení turnaje. Zároveň ale bude možné generovat párování pro následující kola a zobrazit průběžné i konečné výsledky.

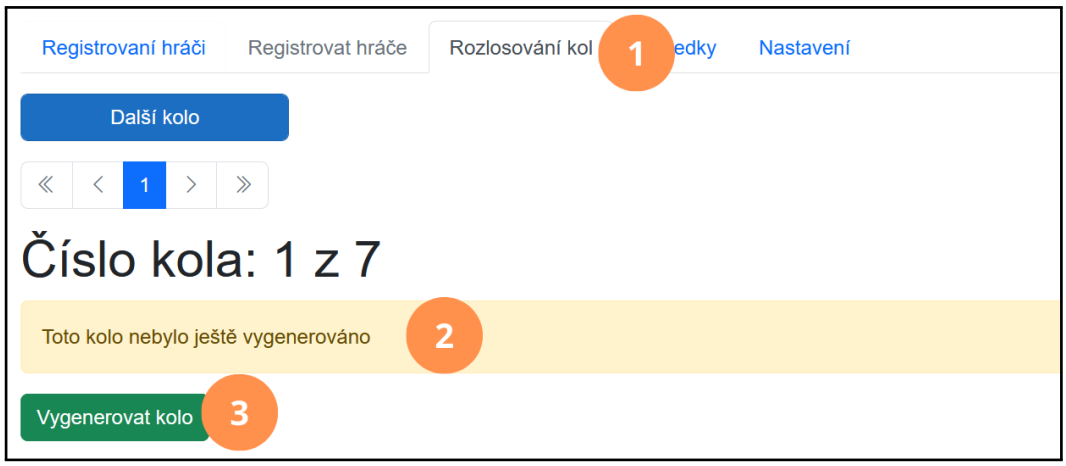
Pokud by naopak došlo k organizačním změnám, je možné turnaj před jeho zahájením odstranit. Odstranění je nevratné a provádí se tlačítkem *Smazat turnaj* (2).



Obrázek 5.13 Správa turnaje - zahájení a smazání

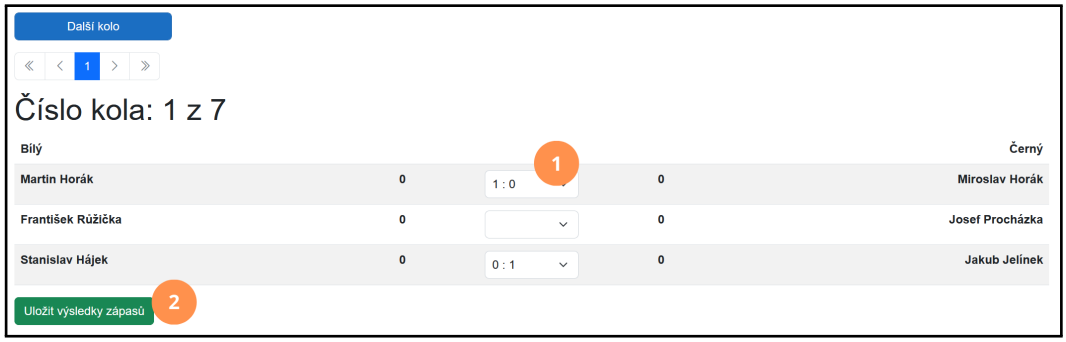
Párování a výsledky zápasů

Na záložce *Rozlosování kol* (1) nalezneme párování pro jednotlivá kola. V případě, že párování ještě nebylo vygenerováno, je nám tato skutečnost oznámena zprávou na oranžovém pozadí (2). Párování kola je možné vygenerovat kliknutím na tlačítko *Vygenerovat kolo* (3), což spustí párovací proceduru a vše proběhne automaticky dle zvoleného typu turnaje.



Obrázek 5.14 Správa turnaje - nevygenerované párování

V případě již existujícího párování jsou zde zobrazeny zápasy, které je nutné v dané kole odehrát. Po skončení jednoho čí více zápasů je možné zvolit jejich výsledky (1) a uložit je (2). Tlačítko pro uložení je viditelné pouze pokud došlo k nějaké změně výsledků.



Obrázek 5.15 Správa turnaje - vygenerované párování

Pokud v turnaji pro týmy nastane situace, že výsledek zápasu dvou týmů bude nerozhodný a turnaj neumožňuje remízu – například *PlayOff* turnaje – je na nás jako organizátorovi rozhodnout o vítězi. Toto rozhodnutí je plně v naší kompetenci a může být provedeno například rozhodujícím zápasem dvou hráčů z obou týmu nebo losem. Informace o vítězi se poté zaznamená u položky *TieBreak* u příslušného zápasu.

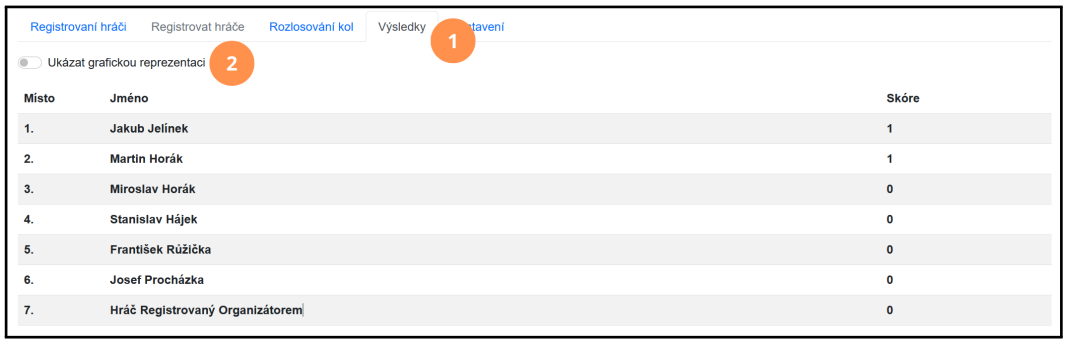
Pro vytvoření nového kola klikneme na tlačítko *Další kolo*, čímž dojde k vytvoření nového kola. Postup generování a ukládání výsledků je stejný jako již bylo popsáno v odstavcích výše. Je nutné poznamenat, že tlačítko *Další kolo* není vždy aktivované. Tlačítko je dostupné za těchto podmínek:

- Jsou vyplněny a uloženy všechny výsledky zápasů - neplatí pro turnaj *Round Robin*

- Aktuálně je zobrazeno poslední kolo
- Jsou vygenerovány zápasy kola, které se musí odehrát.
- Je vyplněn Tiebreak u zápasů týmů, které skončily remízou - pouze turnaje pro týmy

Výsledky turnaje

Na záložce *Výsledky* (1) nalezneme aktuální výsledky turnaje. V základním zobrazení vidíme seznam hráčů či týmů dle aktuálního pořadí. Pro turnaje *Round Robin* máme možnost si výsledky lépe interpretovat za pomocí grafického zobrazení, které se zobrazí po přepnutí přepínače s popiskem *Ukázat grafickou reprezentaci* (2). Pro *Round Robin* turnaje se zobrazí tabulka, kde lze vidět výsledky všech zápasů. V případě *PlayOff* turnaje je zde zobrazený pavouk celého turnaje.



Obrázek 5.16 Správa turnaje - výsledky

5.3 Základní činnosti

Po představení hlavních stránek v aplikaci by již znalejším hráčům či organizátorům mělo být jasné jak jednotlivé činnosti provádět. Ostatní zde naleznou návod jak provádět nejdůležitější činnosti, které aplikace nabízí. Nejprve začneme nastaveném svého účtu a profilu, poté si představíme praktické činnosti.

5.3.1 První spuštění aplikace

Aplikace umožňuje prohlížení jednotlivých turnajů i bez registrace. Abychom ale mohli využít všech funkcí, které aplikace nabízí, je nutné si vytvořit uživatelský profil a poté být při práci s aplikací pod tímto profilem přihlášeni. Pokud nejsme přihlášení, jsme omezeni pouze na pasivní prohlížení dat bez možnosti provádění dalších akcí.

Základem je si vytvořit uživatelský profil. Kliknutím na záložku *Registrace* v hlavním panelu se zobrazí stránka s možností vytvoření účtu vyplněním našeho e-mailu a hesla. Registrace bude dokončena kliknutím na tlačítko *Registrovat se*. Na vyplněnou e-mailovou adresu bude odeslána zpráva s potvrzovacím odkazem, na který je nutné kliknout, aby došlo k potvrzení registrace.

Většina akcí vyžaduje znát naše osobní informace a nebudou pro nás k dispozici bez vyplnění těchto informací na našem profilu. Kliknutím na záložku *Můj profil* v hlavním panelu a následným kliknutím na tlačítko *Upravit profil* dojde k zobrazení formuláře. V něm je možné vyplnit potřebné údaje či nahrát vlastní fotku. Poté je náš účet již plně připraven k užívání bez omezení.

V případě, že již máme dokončenou registraci a došlo k odhlášení, je nutné pro další využívání programu nutné opětovné přihlášení, které lze provést na záložce *Přihlášení*. Poté už je opět možné využívat všechny funkcionality programu.

5.3.2 Funkce programu určené pro hráče

Registrace na turnaj

Jak již bylo zmíněno, turnaje můžeme vyhledávat v sekci *Turnaje*. Vyplníme údaje ve formuláři, které by turnaje, do kterých bychom se chtěli registrovat, měly splňovat. Po kliknutí na tlačítko *Vyhledat* se zobrazí turnaje splňující daná kritéria. Detailní informace o turnaji je možné vidět na stránce zobrazené po kliknutí na tlačítko *Zobrazit detaily*. Kliknutím na tlačítko *Registrovat se* u konkrétního turnaje dojde k zobrazení stránky s registrací na turnaj. V případě, že se jedná o turnaj pro jednotlivce zobrazí se nám okamžitě informace o tom, zda je registrace možná. Pokud se jedná o turnaj pro týmy, zobrazí se nám seznam týmů, které spravujeme (viz Funkce programu určené pro hráče - pokročilé). Pokud je registrace možná, je nám tato informace sdělena zprávou se zeleným pozadím. Kliknutím na potvrzovací tlačítko dojde registraci. Nyní už pouze zbývá se těšit na nadcházející turnaj.

Zobrazení přihlášek

Pokud jsme již provedli registraci na nějaký turnaj, můžeme jej vidět na záložce *Moje turnaje*. Zde vidíme všechny turnaje, které nás následující dny i týdny čekají. Zobrazeny jsou i již ukončené turnaje, ve kterých jsme hráli. V případě, že se turnaje nemůžeme účastnit, provedeme odhlášení z turnaje pomocí tlačítka *Zrušit registraci*.

Průběh turnaje

Pokud turnaj již probíhá můžeme v aplikaci sledovat aktuální rozlosování kol či výsledky. Na záložce *Moje turnaje* můžeme u daného turnaje kliknutím na tlačítko *Zobrazit detaily* zobrazit detaily o turnaji. Zde v jednotlivých záložkách můžeme vidět rozlosování pro jednotlivá kola, registrované hráče či výsledky.

5.3.3 Funkce programu určené pro hráče - pokročilé

V případě, že bychom se chtěli účastnit týmového turnaje, musíme nejprve tým vytvořit. Na záložce *Spravované týmy* v hlavním panelu vidíme všechny své týmy, které jsme vytvořili. Kliknutím na tlačítko s ikonkou „+“ se zobrazí stránka, kde je možné vytvořit nový tým. Tento tým poté můžeme ze stránky se všemi týmy zobrazit v detailu. Kde pomocí šipek můžeme měnit pořadí členů v týmu a zároveň přidávat nové hráče. Také zde vidíme turnaje, na které je tým přihlášen.

Po vytvoření a nastavení týmu již zbývá pouze vyhledat turnaj na záložce *Turnaje* v hlavním panelu, na který bychom chtěli tým registrovat (je možné filtrovat pouze turnaje pro týmy). Proces registrace je totožný jako v případě registrace jednotlivců.

5.3.4 Funkce pro organizátory - vytvoření a průběh turnaje

Pokud bychom chtěli uspořádat vlastní šachový turnaj, je nutné ho nejprve vytvořit. Na záložce *Vytvořit nový turnaj* v hlavním panelu vyplníme všechny vyžadované údaje a klikneme na tlačítko *Vytvořit*. Tímto se turnaj vytvoří a dojde k zobrazení potvrzovací zprávy. Je ale nezbytné provést další nastavení, aby turnaj mohl správně proběhnout. Klikneme na tlačítko *Na stránku se správou turnaje*, čímž nám bude zobrazena stránka, kde lze provádět veškeré akce související se správou turnaje (podrobné informace v podkapitole *Záložka Správa turnaje*). Nejprve v nastavení nastavíme období, kdy se lze na turnaj přihlašovat, kapacitu a případně další informace. Pokud pořádáme turnaj pro týmy nezapomeneme nastavit velikost týmů, pro které je turnaj určen. Všechny turnaje, které jsme takto vytvořili, vidíme na záložce *Spravované turnaje* v hlavním panelu.

V den konání turnaje si můžeme na stránce se správou turnaje zobrazit dosud přihlášené hráče a případně odebrat ty, kteří se nedostavili. Pokud dorazil někdo, kdo nebyl dopředu registrován, můžeme jej přidat v sekci *Registrovat hráče*. Poté již můžeme zahájit turnaj kliknutím na tlačítko *Zahájit turnaj*. Od této chvíle není možné provádět změny v registracích hráčů. Pro rozlosování prvního kola klikneme v sekci *Rozlosování kol* na tlačítko *Vygenerovat kolo*. Po chvíli se nám zobrazí vygenerované zápasy. Až budou známy výsledky, můžeme je nastavit u daných zápasů a uložit je. Po odehrání všech zápasů kola klikneme na tlačítko *Další kolo* a vygenerujeme následující kolo.

Výsledky můžeme zobrazit kdykoliv během turnaje i po jeho skončení v sekci *Výsledky*.

Závěr

Tato práce si kladla za cíl vytvořit program, který by usnadnil průběh šachového turnaje pro hráče i organizátory. Abychom dosáhli tohoto cíle, tak jsme nejprve provedli analýzu činností, které organizátoři i hráči musí během turnaje provádět. Poté jsme analyzovali kvality i nedostatky současných programů určených k organizování šachových turnajů. Získané znalosti jsme využili při vývoji aplikace. Můžeme konstatovat, že výsledná webová aplikace usnadňuje organizátorům správu šachového turnaje. Díky naší aplikaci odpadlo hráčům i organizátorům zdlouhavé opakované zadávání údajů při registraci. Též se zjednodušil přístup k rozlosování jednotlivých kol i ke konečným výsledkům.

Při implementaci aplikace jsme se poučili z chyb autorů současných programů. Vznikl tedy funkční program, který dokáže ovládat prakticky každý. Program je též přístupný pro všechny, jelikož k používání stačí pouze webový prohlížeč, což dělá aplikaci přístupnou i pro lidi, kteří neumí ovládat počítač na pokročilé úrovni.

Záležet jsme si dali i na uživatelském rozhraní, aby působilo přirozeným dojmem a poskytovalo dostatek zpětné vizuální vazby. Zároveň jsme vybrali dobře rozlišitelné barvy pro jednotlivé prvky a akce.

Program v současnosti obsahuje všechny důležité funkcionality. Pro zlepšení uživatelského komfortu je ale možné implementovat následující funkce:

- **Propojení s FIDE** - Informace o hráčích registrovaných v organizaci FIDE by bylo vhodné automaticky stahovat z databáze organizace a pomocí nich aktualizovat údaje v našem programu.
- **Automatické vyplnění adresy** - Různé mapové služby by nám umožnili automatické doplňování adresy dle již vyplněných informací a zároveň by bylo možné vyplněné adresy validovat.
- **Chat a notifikace** - Pro akce, které nejsou v aplikaci podporované, by bylo vhodné implementovat chat, kde by si mohli lidé napsat o různých organizačních záležitostech. Též by uživatelský zážitek zlepšilo oznámení o novém kole či samotném zahájení turnaje.

I bez těchto možných vylepšení je ale program užitečným pomocníkem na organizaci šachového turnaje a snad v budoucnu usnadní mnoha lidem jejich cestu organizátorů šachových turnajů.

Literatura

1. Organization, FIDE. *General Rules and Technical Recommendations for Tournaments / 04. FIDE Swiss Rules* [online]. [cit. 2024-05-03]. Dostupné z: https://handbook.fide.com/chapter/C0401.
2. Organization, FIDE. *C.04.1 Basic rules for Swiss Systems* [online]. [cit. 2024-05-05]. Dostupné z: https://spp.fide.com/wp-content/uploads/dutch2017-approved.pdf.
3. Organization, FIDE. *C. General Rules and Technical Recommendations for Tournaments / 07. Tie-Break Regulations* [online]. [cit. 2024-05-03]. Dostupné z: https://handbook.fide.com/chapter/TieBreakRegulationsPre2023.
4. Organization, FIDE. *FIDE LAWS of CHESS* [online]. [cit. 2024-04-15]. Dostupné z: https://www.fide.com/FIDE/handbook/LawsOfChess.pdf.
5. Neznámý. *Swiss-Manager order* [online]. [cit. 2024-04-15]. Dostupné z: https://swiss-manager.at/order.aspx.
6. Neznámý. *Swiss-Manager User's Guide* [online]. [cit. 2024-05-05]. Dostupné z: https://swiss-manager.at/unload/swiss_manager_user_guide.pdf.
7. Suits, Thad. *Ordering and upgrading* [online]. [cit. 2024-04-15]. Dostupné z: https://swisssys.com/ordering.html.
8. Software, SwissSys. *Swiss-Manager User's Guide* [online]. [cit. 2024-05-05]. Dostupné z: https://swisssys.com/.
9. VEKTRA spol. s.r.o. *Swips* [online]. [cit. 2024-05-05]. Dostupné z: https://www.swips.eu/.
10. Neznámý. *What is Free-Swiss?* [online]. [cit. 2024-04-17]. Dostupné z: https://www.free-swiss.com/.
11. Group, The PHP. *History of PHP* [online]. [cit. 2024-04-15]. Dostupné z: https://www.php.net/manual/en/history.php.php.
12. Swacha J.; Kulpa, A. *Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks* [online]. [cit. 2024-04-17]. Dostupné z: https://doi.org/10.3390/electronics12173563.
13. Roth, Daniel. *Blazor now in official preview!* [online]. [cit. 2024-04-15]. Dostupné z: https://devblogs.microsoft.com/dotnet/blazor-now-in-official-preview/.
14. Migla, Kaspars. *Rating analytics: The number of rated chess players goes up* [online]. [cit. 2024-04-17]. Dostupné z: https://www.fide.com/news/288.
15. Harpreet Kaur Jaspreet Kaur, Kamaljit Kaur. A Review Of Non Relational Databases, Their Types, Advantages And Disadvantages. *International journal of engineering research and technology.* 2013.