FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém na podporu organizace šachových turnajů
Student:	Jana Maříková
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Při šachových turnajích je třeba řešit organizační náležitosti, kterými jsou například: evidence hráčů, rozlosování hráčů, zápis výsledků, formát her atd. Cílem této práce je realizace podpůrné webové aplikace.

Při realizaci postupujte podle těchto kroků:

1. Analyzujte potřeby cílové skupiny uživatelů.
2. Na základě analýzy proveďte vhodný softwarový návrh.
3. Při návrhu řádně diskutujte budoucí funkčnost aplikace s potenciálními uživateli aplikace.
4. Implementujte za použití vhodných technologií první verzi aplikace.
5. Implementaci podrobte vhodným testů. Vhodnost testů řádně podložte.
6. Funkční aplikaci nasaďte a zajistěte možnost použití, alespoň uzavřené skupině uživatelů.
7. Zhodnoťte možnosti rozšíření aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 25. prosince 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Systém na podporu organizace šachových turnajů

Jana Maříková

Katedra softwarového inženýrství
Vedoucí práce: Ing. Filip Glazar

15. května 2019

Poděkování

Tímto bych chtěla upřímně poděkovat Ing. Filipu Glazarovi za jeho čas, cenné rady a pomoc při realizaci této bakalářské práce. Děkuji také svým kamarádům, kteří se ochotně podíleli na uživatelském testování. Velký dík patří mé rodině, která mě podporovala a motivovala v průběhu celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 15. května 2019 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2019 Jana Maříková. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Maříková, Jana. *Systém na podporu organizace šachových turnajů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019. Dostupný také z WWW: ⟨`http://www.tnchessmanager.cz/`⟩.

Abstrakt

Tato práce se zabývá návrhem a implementací systému na organizaci šachových turnajů. Důraz je kladen především na losování typu round-robin a švýcarský. Nejprve je rozebrána daná problematika, následně jsou analyzovány již existující programy a cílová skupina uživatelů. Na základě toho jsou vytvořeny požadavky na systém a návrh aplikace.

Klíčová slova šachy, párovací program, švýcarský systém, round-robin, React, Spring.

Abstract

Subject of this bachelor thesis deals with design and implementation of system for chess tournaments organization. Chess pairing systems round-robin and swiss are in the main focus. The problematic of chess tournament organization is describe, then the existing programs and potencional users are analyzed. Based on that functional, nonfunctional and use cases are listed.

Keywords chess, pairing program, swiss system, round-robin, React, Spring.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

Šachy jsou jedna z nejstarších a nejznámějších deskových her na světě. První turnaj moderních šachů se odehrál v Leeds v roce 1841. O deset let později se pak konal první mezinárodní turnaj v Londýně. Od té doby přibývá šachových hráčů a ruku v ruce s tím roste i počet turnajů. Jen v České republice je kolem patnácti tisíc registrovaných aktivních hráčů a ročně se pořádají stovky turnajů různých velikostí a úrovní. Právě na organizaci turnajů jsem se rozhodla ve své bakalářské práci zaměřit.

Významným problémem při pořádání turnaje je především registrace hráčů, losování jednotlivých kol a určení pořadí hráčů.

Před nástupem informačních technologií se veškerá administrace spojená s organizací turnajů musela dělat ručně. Aktuálně existuje několik programů, které se snaží organizátorům s větším či menším úspěchem tyto úkoly usnadnit. Většina těchto programů je zastaralých, nejsou intuitivní a zejména při menších klubových turnajích je nevýhodou i to, že bývají zpoplatněny.

Cílem této bakalářské práce je analyzovat existující řešení a cílovou skupinu uživatelů. Na základě toho vymezit požadavky na systém, který bude následně navržen a za použití zvolených technologií naimplementován a řádně otestován.

Práce je členěna do sedmi kapitol. V kapitole 1 rozebírám šachovou problematiku z pohledu organizace turnajů a představuji dva losovací systémy, kterými se budu v práci zabývat. Následuje kapitola 2, ve které představuji již existující programy na podporu organizace turnajů a cílovou skupinu uživatelů. V kapitole 3 specifikuji požadavky na systém a návrhuji základní architekturu systému. Dále v kapitolách 4 a 5 představuji použitou technologii s ukázkami částí systému, které byly zajímavé buď z hledistka návrhu nebo realizace. Testování systému popisuji v kapitole 6 a práce zakončuji kapitolou 7, která nabízí možnosti rozšíření práce.

Kapitola 1

Úvod do problematiky šachových turnajů

Pro pochopení následujících kapitol je nejprve nutné vysvětlit některé problematické aspekty organizace šachových turnajů a uvést potřebné pojmy. V této kapitole také představím dva nejčastější losovací systémy a jejich obecné algoritmy, kterými se budu dále zabývat v návrhu a implementaci.

1.1 Losovací systémy

Cílem losovacích systémů je dosáhnout co největší spravedlnosti v párování a co možná nejpřesněji určit oprávněného vítěze turnaje. Při velkém počtu hráčů je to poměrně komplikovaný úkol. Na korektnost používaných systémů proto dohlíží organizace Fédération Internationale des Échecs (FIDE).

FIDE je světová šachová organizace. Tato organizace vydává regule pro šachové turnaje, definuje platné losovací systémy a mnoho dalšího. Je zodpovědná například i za činnosti, jako je výpočet FIDE ela a titulu (význam ela a titulu je popsán v kapitole 1.1.1). Na oficiální turnaje, kde jsou tyto činnosti vyžadovány, je tedy doporučeno používat program, který FIDE schválilo. V republikovém měřítku u nás funguje šachový svaz České republiky, který pod FIDE spadá [1].

Losovací systém turnaje se musí určit před jeho začátkem a většinou se volí podle očekávaného počtu hráčů. Aktuálně se používají především dva systémy turnaje, kterými se budu v této práci zabývat:

- **Round-robin systém** – systém každý s každým, více v kapitole 1.2.
- **Švýcarský systém** – systém používaný především u větších turnajů, více v kapitole 5.3.

1.1.1 Startovní listina

Před začátkem losování prvního kola turnaje je nutné sestavit startovní listinu hráčů. Podle FIDE pravidel pro švýcarský systém [2] je pořadí hráčů ve startovní listině určeno podle následujících kritérií.

1. **Elo** – určuje šachovou sílu hráče. Každý hráč může mít několik typů ela: blitz, rapid, FIDE a národní elo. Před turnajem se vždy určí, jaké elo nebo ela se budou brát v potaz. Žádné z uvedených el nemůže být nižší než 1 000. Pouze v případě, že hráč žádné elo nemá, uvádí se hodnota 0. [3] Pro představu, nejsilnější hráč světa má FIDE elo 2 845. [4]

2. **Titul** – určuje *úroveň*, které hráč dosáhl. Pro jeho získání je potřeba splnit určitá definovaná kritéria. Jakmile hráč jednou titul získá, už ho nemůže ztratit. Eviduje se vždy pouze titul s nejvyšší váhou. Tituly jsou seřazeny sestupně podle váhy: Grand Master (GM), Woman Grand Master (WGM), International Master (IM), Woman International Master (WIM), Fide Master (FM), Woman Fide Master (WFM), Candidat Master (CM), Woman Candidat Master (WCM). Existují i další tituly, ty se ale většinou neuvádějí.

3. **Abeceda** – pokud mají hráči stejná ela a stejný titul, řadí se podle abacedního pořadí jejich příjmení.

Podle této listiny se hráčům přiřadí startovní čísla. [2]

1.1.2 Bye

Bye je definován jako nehraná partie. Může jít například o situaci, kdy je v turnaji lichý počet hráčů nebo se některému hráči nedostaví soupeř k partii. Nastává otázka, jak přistupovat k *bye*, ať z hlediska bodů, barvy, pomocného hodnocení atd. Proto jsou definována následující pravidla.

Pro účely výpočtu pomocného hodnocení se *bye* bere jako virtuální oponent, který měl před začátkem kola stejný počet bodů jako hráč P, se kterým se utkal. Každé další kolo pak remizoval. Za parii pak dostane hráč tolik bodů, kolik bylo definováno před začátkem turnaje a žádnou barvu. Z toho vyplývá vzorec: [5]

$$Svon = s + (1–f) + 0,5 * (n–r) \tag{1.1}$$

- $Svon$ – počet bodů virtuálního oponenta po kole n.
- s – body hráče P před kolem r.
- f – body získané proti virtuálnímu oponentovi (definované na začátku turnaje).
- r – kolo ve kterém se utkal s hráčem P.

1.1.3 Pomocné hodnocení

Pomocné hodnocení se používá pro určení pořadí hráčů, kteří dosáhli stejného počtu bodů a je tedy nutné jejich pořadí určit podle jiných kritérií. Následující pomocná hodnocení jsou schválena organizací FIDE, viz [5].

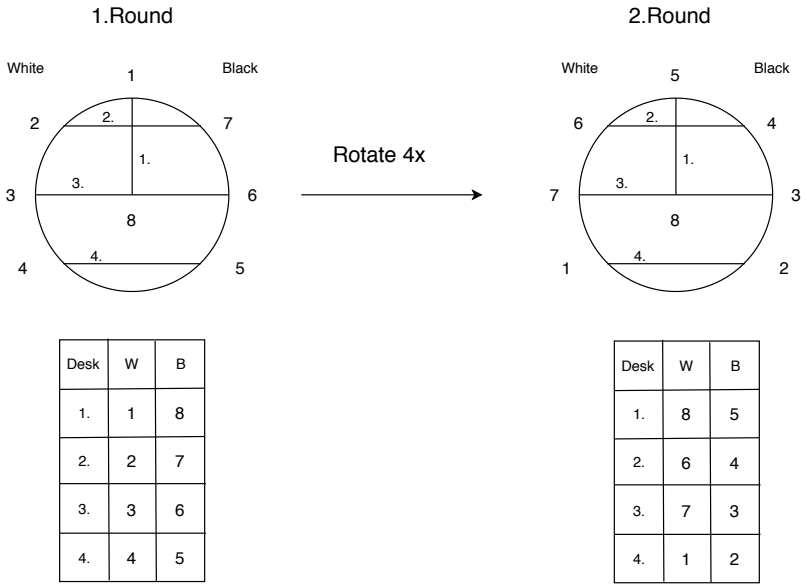
- **Přímé střetnutí** – výsledek partie, kterou spolu hráči se stejným počtem bodů sehráli.
- **Průměrné elo soupeřů** – součet el soupeřů vydělené jejich počtem. Musí se vždy jednat o stejný typ ela.
- **Buchholz** – součet bodů soupeřů.
- **Střední buchholz** – součet bodů soupeřů bez nejlepšího a nejhoršího hráče.
- **Koya** – počet bodů získaných proti hráčům, kteří získali 50 % bodů a více.
- **Rozšířená koya** – počet bodů získaných proti hráčům, kteří získali 50 % bodů a více bez n nejhorších a n nejlepších hráčů.
- **Počet her s černými figurami** – berou se v potaz pouze sehrané partie.
- **Sonneborn-Berger** – součet bodů soupeřů, které hráč porazil a polovina bodů soupeřů, proti kterým hráč remízoval.
- **Počet výher** – počet výher proti reálnému oponentovi. Bod získaný bez hraní se tedy nepočítá.

Pro každý systém existuje sada doporučených pomocných hodnocení. Záleží však čistě na organizátorech turnaje, zda se daného doporučení budou držet, nebo si zvolí jinou sadu pomocných hodnocení. [5]

1.1.4 Výsledky

Výsledky jsou vždy ve formátu výsledek bílého – výsledek černého. Kromě klasických výsledků (1–0, 0–1, ½–½) se mohou vyskytnout i speciální výsledky. Z nich se nejčastěji používají 1F–0F, 0F–1F a 0F–0F. Písmeno F u výsledku je první písmeno slova *forfeith*, v českém překladu *propadnutí*. Tyto výsledky značí fakt, že hráč, který obdržel nulu, prohrál kontumačně. Hráč je zkontumován, pokud se nedostaví k partii, zazvoní mu telefon nebo se jinak prohřeší proti pravidlům.

Speciální výsledky mohou mít vliv na losování dalších kol turnaje a na výpočet pomocného hodnocení. Proto je nezbytné je rozlišovat od klasických výsledků.



Desk	W	B
1.	1	8
2.	2	7
3.	3	6
4.	4	5

Desk	W	B
1.	8	5
2.	6	4
3.	7	3
4.	1	2

Obrázek 1.1: Round-robin

1.2 Round-robin systém

Round-robin je typický systém každý s každým. Pokud je v turnaji použit tento systém, musí být všechna kola zveřejněna najednou.

Existuje více způsobů, jak ho losovat. FIDE doporučuje řídit se Bergrovými tabulkami. [6]

1.2.1 Bergrovy tabulky

Algoritmus pro jednoduché sestrojení Bergrových tabulek vymyslel matematik *Édouard Lucas* [7]. Na obrázku 1.1 je daný algoritmus zobrazen pomocí grafu doplněného tabulkami. Tento graf je určen pro turnaj s 8 nebo 7 hráči. V příloze B jsou pak přiloženy Bergrovy tabulky až pro 12 hráčů, kde jsou zobrazena všechna kola.

Čísla na grafu představují startovní čísla hráčů. Pokud je v turnaji lichý počet hráčů, místo hráče se starovním číslem osm je přiřazen *bye*.

Každá dvě čísla spojuje hrana, ta určuje, kteří hráči spolu dané kolo budou hrát. Hrany jsou označeny číslem, které určuje na kolikáté šachovnici se hráči utkají. Každé kolo se hráči posunou o čtyři pozice ve směru hodinových ručiček. To proběhne přesně sedmkrát. Pouze hráč s nejvyšším startovním číslem se neposouvá.

Bílou barvu pak mají hráči, kteří jsou na levé straně grafu. U hráče s nejvyšším startovním číslem se barva pravidelně střídá a začíná vždy s černou barvou.

1.3 Švýcarský systém

Švýcarský systém je určen pro turnaje s více hráči, kde není časové možné, aby hrál každý s každým. Existuje více typů švýcarských systémů schválených světovou šachovou organizací FIDE. Všechny tyto systémy dodržují základní pravidla převzatá z legislativy šachového svazu ČR, viz [8].

1. Počet kol turnaje je zveřejněn předem.
2. Dva hráči se spolu neutkají více něž jednou.
3. Pokud měl hráč volno, nemá nárok na další volno.
4. Hráči se stejným bodovým hodnocením hrají proti sobě, pokud to lze.
5. Hráč dostane tu barvu, kterou měl méně často, pokud to lze.
6. Hráč dostane opačnou barvu než měl v posledním kole, pokud to lze.
7. Konečné pořadí je určeno počtem bodů.

1.3.1 Holandský systém

Holadský systém je nejpoužívanější typ švýcarského systému. Typy švýcarského systému se většinou liší v tom, kdo má prioritu při nalezení vhodného soupeře. U holandského systému je priorita určena početem bodů. Níže uvedený zjednodušený algoritmus spolu s definovanými termíny vychází z příkladu organizace turnaje holandským systémem popsaného *Mario Heldem*, viz [9].

Použité termíny

Před uvedením kroků algoritmu k losování holandského systému je potřeba definovat použité termíny.

- **Homogenní skupina** – všichni hráči mají stejný počet bodů.
- **Heterogenní skupina** – všichni hráči nemají stejný počet bodů.
- **Barevná preference** – rozdíl mezi počtem her bílými a černými barvami.
  - Střední barevná preference (rozdíl je roven 0).
  - Silná barvná preference (rozdíl je roven 1 nebo -1).
  - Absolutní barevná preference (rozdíl je mimo interval $\langle -1, 1 \rangle$).
- **Top hráč** – hráč, který má více než 50% zisk bodů.
- **Absolutní párovací kritéria** – základní pravidla švýcárského systému.
- **Silné párovací kritérium** – dva hráči se stejnou absolutní barevnou preferencí nebudou hrát proti sobě.

1.3.2 Algoritmus pro losovaní holandského systému

Hráči se rozdělí do skupin podle bodů a v každé skupině se seřadí podle startovních čísel. Na každou skupinu od nejsilnější po nejslabší se aplikují následující kroky.

Krok 1: Kontrola bodové skupiny

Pokud ve skupině existuje hráč, který nemá v dané skupině žádného validního protivníka, bez toho aniž by byla porušena absolutní nebo silné párovací kritéria, je hráč přesunut do nejbližší nižší bodové skupiny.

- Pokud se jedná o nejslabší bodovou skupinu pokračuje se od kroku 9.
- Pokud již hráč/i byli přesunuti ze silnější bodové skupiny, vrací se zpět do předchozí bodové skupiny a je snaha nalézt jiné hráče k přesunutí.

Krok 2: Výpočet parametrů

Výpočet parametrů slouží k sestavení párovacích kritérií.

- P_0 = počet hráčů v bodové skupině/2. $P_1 = P_0$.
- M_0 = počet hráčů, kteří byli přesunuti z vyšší bodové skupiny. $M_1 = M_0$.
- X_1 = počet párů, který může být sestaven i když nevyhovuje barevné preferenci.

Krok 3: Sestavení párovacích kritérií

Párovací kritéria představují seznam požadavků, které musí páry splnit, aby byly akceptovány.

1. P = počet párů, které musí být vytvořeny z bodové skupiny.
   - Homogenní skupina: $P = P_1$.
   - Heterogenní skupina: $P = M_1$.
2. Dva hráči se stejnou absolutní barevnou preferencí spolu nesmí hrát.
3. V sudém kole se přistupuje k hráčům se silnou barevnou preferencí jako k hráčům s absolutní barevnou preferencí.
4. Počet párů, které se musí sestavit je roven X_1.
5. Žádný hráč nesmí hrát s hráčem, který má nižší nebo vyšší počet bodů dvakrát po sobě.
6. Žádný hráč nesmí hrát s hráčem, který má vyšší počet bodů, pokud před dvěmi koly hrál s hráčem, který má vyšší počet bodů.

Krok 4: Rozdělení hráčů do podskupin

V případě homogenní skupiny se hráči rozdělí na poloviny. První polovina se přiřadí podskupině S_1, druhá podskupině S_2. Pokud se jedná o heterogenní skupinu, tak hráči, kteří mají více bodů se přidělí do podskupiny S_1, ostatní pak do skupiny S_2. Hráči v podskupinách S_1 a S_2 se seřadí podle: 1. bodů, 2. startovního čísla.

Krok 5: Párování hráčů

Nejsilnější hráč ze skupiny S_1 se spáruje s nejsilnějším hráčem z S_2. Pokud je získáno P párů a jsou splněna všechna kritéria z bodu 3. jsou dané páry akceptovány. Pokud v dané skupině zbyli nespárovaní hráči tak:

- Homogenní skupina – ze zbylých hráčů se vytvoří samostatná skupina a pokračuje se od kroku 1.
- Heterogenní skupina – zbylí hráči se přesunou do další nižší bodové skupiny a předefinuje se $P = P_1 - M_1$ a pokračuje se od kroku 4.

Pokud páry nesplňují požadavky z kroku 3 je aplikován krok 6.

Krok 6: Permutace

Změna pořadí hráčů ve skupině S_2. Pro tento účel se vybírá další nejnižší možná permutace. Pokud již nelze provést žádná permutace a páry stále nevyhovují kritériím z kroku 3, aplikuje se výměna, tedy krok 7.

Krok 7: Výměna

Výměna hráče ze skupiny S_1 s hráčem ze skupiny S_2. Je snaha vybrat takové hráče, kteří jsou si svým startovním číslem nejblíže. Za předpokladu, že byla provedena výměna a hráči stále neodpovídají požadavkům, pokračuje se v kroku 5. Pokud byly provedeny všechny možné výměny následovány permutacemi, aplikuje se další krok.

Krok 8: Relaxace párovacích kritérií

Do tohoto kroku se dostane, pokud permutace ani výměna nevedla k uspokojujícím výsledkům. Postupně se ruší nebo snižují požadavky z kroku 3. Po každém snížení nebo zrušení požadavku se opět aplikuje krok 4.

Krok 9: Párování nejslabší bodové skupiny

V případě homogenní skupiny se vrací zpět do předposlední skupiny a je snaha o nalezení jiných párů, aby šla spárovat poslední bodová skupina. Pokud se to nepodaří, sloučí se předposlední a poslední bodová skupina. V případě heterogenní skupiny je snaha snížit počet hráčů přesunutých z vyšší bodové skupiny tím, že se sníží parametry, viz krok 10.

Krok 10: Snížení parametrů

- Homogenní skupina
  - pokud $P_1 > 0$, sníží se P_1 o 1 a pokračuje se od 3. kroku.
  - pokud $P_1 = 0$, je celá bodová skupina přesunuta do nejbližší nižší a pokračuje se od kroku 1.
  - pokud $X_1 > 0$, sníží se X_1 o 1 a pokračuje se od 3. kroku.
- Heterogenní skupina
  - pokud $M_1 > 1$, sníží se M_1 o 1 a pokračuje se od 3. kroku.

Pokud má každý hráč validního protivníka, přiřadí se každému páru číslo šachovnice a barva.

Kapitola 2

Průzkum

V této kapitole se zabývám průzkumem, na základě kterého bylo rozhodnuto, zda může nový program na losování šachových turnajů přinést něco nového. V kapitole 2.1 analyzuji již existující programy na podporu organizace šachových turnajů a následně v kapitole 2.2 mapuji cílovou skupinu uživatelů a zjišťuji, zda mají s programy z kapitoly 2.1 nějaké zkušenosti.

2.1 Existující řešení

V počáteční fázi projektu je třeba zjistit, jestli už obdobné programy neexistují a jaké jsou jejich výhody a nevýhody. V současné době existuje několik programů sloužících k podpoře organizace šachových turnajů. Programy jsem posuzovala dle několika aspektů:

- **Aktuálnost** – jak je program starý a zda je stále vyvíjen.
- **Uživatelská přívětivost** – zda je práce s programem intuitivní a funkcionality jsou logicky navrženy.
- **Technická omezení** – zda je program použitelný na více operačních systémech/platformách a zda je potřebný přístup k internetu.
- **Požadavky na systém** – zda program obsahuje všechnu potřebnou funkcionalitu, kterou uživatel využije.
- **Chyby** – jestli program obsahuje nějaké chyby, které by mohly mít negativní vliv na průběh turnaje.

Pro účely průzkumu jsem zkoumané programy rozdělila do dvou kategorií: desktopové a webové aplikace.

2.1.1 Desktopové aplikace

Značnou výhodou desktopových aplikací je, že nepotřebují neustálý přístup k internetu a dají se tedy používat offline. Na druhou stranu pokud se používá program, který není integrován s webovými stránkami nebo organizátoři tuto funkčnost nevyužívají, je pro hráče poměrně problematické zjišťování výsledků a rozlosování. Dle mých zkušeností se to většinou řeší tak, že se po každém kole dané listiny vytisknou a vyvěsí v hrací místnosti. To ale není úplně ideální řešení pro turnaje, které se hrají déle než jeden den a hráči tedy nejsou v hrací místnosti neustále přítomni.

Všechny desktopové aplikace, které jsou níže rozebrány, mají licenci FIDE. Tyto programy jsou primárně určeny pro oficiální turnaje se zápočtem na FIDE elo. Licenci FIDE má v dnešní době pět desktopových programů: *Vega*, *SwissSys*, *Swiss Manager*, *Swiss Master* a *Swiss-chess* [10].

Jedna z podmínek pro zisk licence FIDE je, že program musí poskytovat zkušební verzi. Tato zkušební verze většinou představovala omezení v počtu hráčů v turnaji nebo počtu kol, ale základní funkcionalita zůstala nezměněná. Díky tomu jsem si mohla programy zdrama nainstalovat a vyzkoušet. Protože jsou si zmíněné programy dost podobné, zaměřila jsem se pouze na tři, které mi přišly nejlepší nebo nečím zajímavé a ty podrobněji rozeberu.

Swiss Master

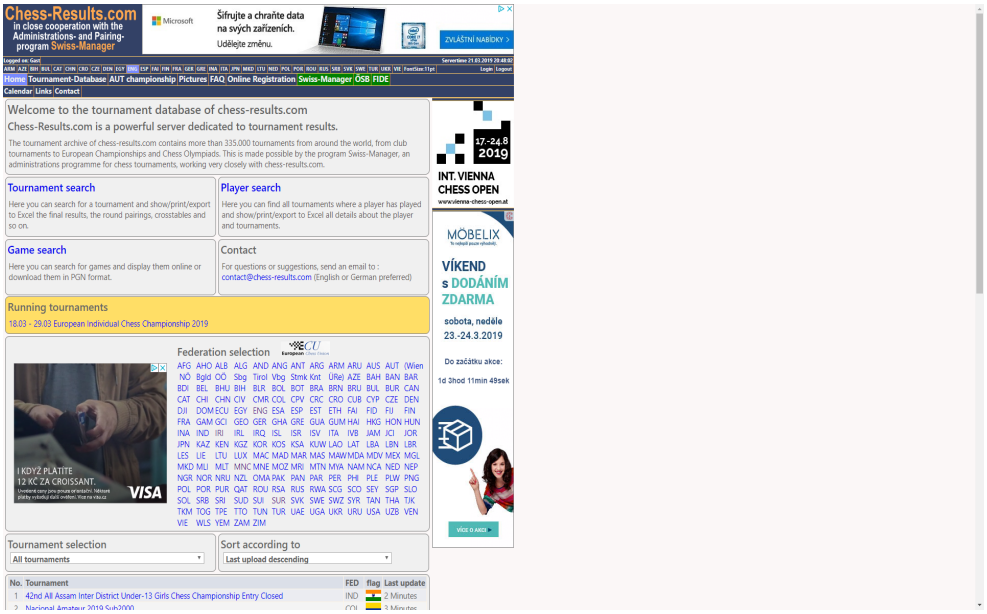
Autor: Royal Dutch Federation
Webová stránka: `www.schaakbond.nl/swissmaster`
Cena: 50 €

Jedná se o vůbec první program na losování šachových turnajů. Podle [11] algoritmus pro holandský systém vznikl právě na základě losovacího algortimu použitého v programu Swiss Master. Všechny ostatní programy používají na losování kol holandského systému externí program JaVaFo (více v kapitole 5.3). Nejnovější verze vznikla v roce 2015, ale vzhledově působí stále zastarale a ovládání není ani trochu intuitivní, při práci s programem jsem musela neustále nahlížet do manuálu.

Swiss Manager

Autor: Heinz Herzog (AUT)
Webová stránka: `www.chess-results.com/`
Cena: Light/Full verze 75/150 €

Jedná se o komerční program, licence byla zakoupena ŠSČR, a proto by neměl být používán oddíly, které nejsou jeho členem.[12] Swiss Manager je integrován se stránkami Chess-results, viz obrázek 2.1. Na těchto stránkách je možnost zveřejnit turnaje, které byly losovány pomocí Swiss Manageru.



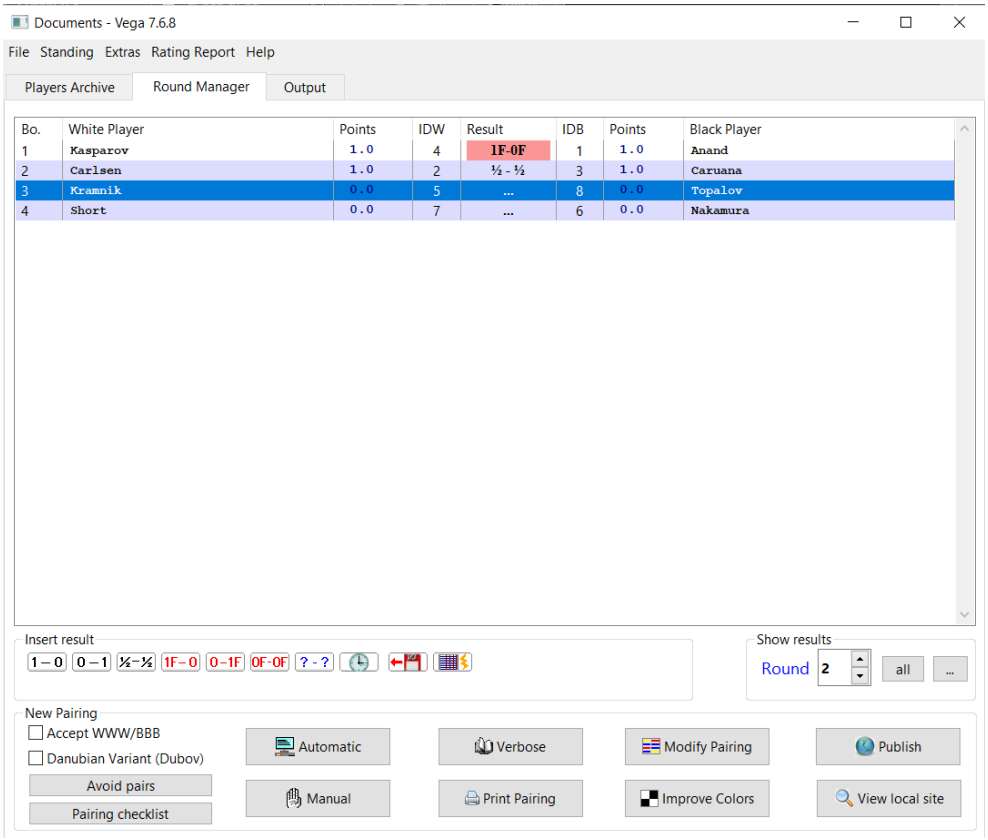
Obrázek 2.1: Chess-results – domovská stránka

Na Chess-results jsou zveřejňována rozlosování kol, pořadí a výsledky, které jsou přístupné všem uživatelům. Aktuálně je zde zveřejněno okolo 100 000 turnajů. [13]

Design programu je poplatný době jeho vzniku a v podstatě není naděje, že by se do budoucna změnil. V současné době je problematická i oprava nalezených chyb:

- **Nefunkční ruční párování hráčů** – ruční párování hráčů se nesmí použít, pokud má být turnaj započten na elo. U nezápočtových turnajů se používá ale dost často, např. pokud se dvá hráči dostaví později.
- **Nefunkční kritéria při lichém počtu hráčů** – chybné výsledky, pokud se jako pomocné hodnocení použije např. průměrný rating soupeřů.
- **Chybné průběžné/konečné pořadí při hodnocení 3-1-0** – i když toto hodnocení lze nastavit, tak je při generování pořadí ignorováno.
- **Práce s ratingovými listinami** – v současnosti se elo hráče počítá pro praktický šach, rapid a bleskový šach. V programu figurují tyto tři ratingy jako samostatné listiny a při výběru hráče se každý hráč zobrazuje vícekrát.

Použití některých dalších vlastností je poměrně krkolomné a práce s programem vyžaduje dost zkušeností. Swiss Manager je spustitelný pro Windows XP a novější [13].



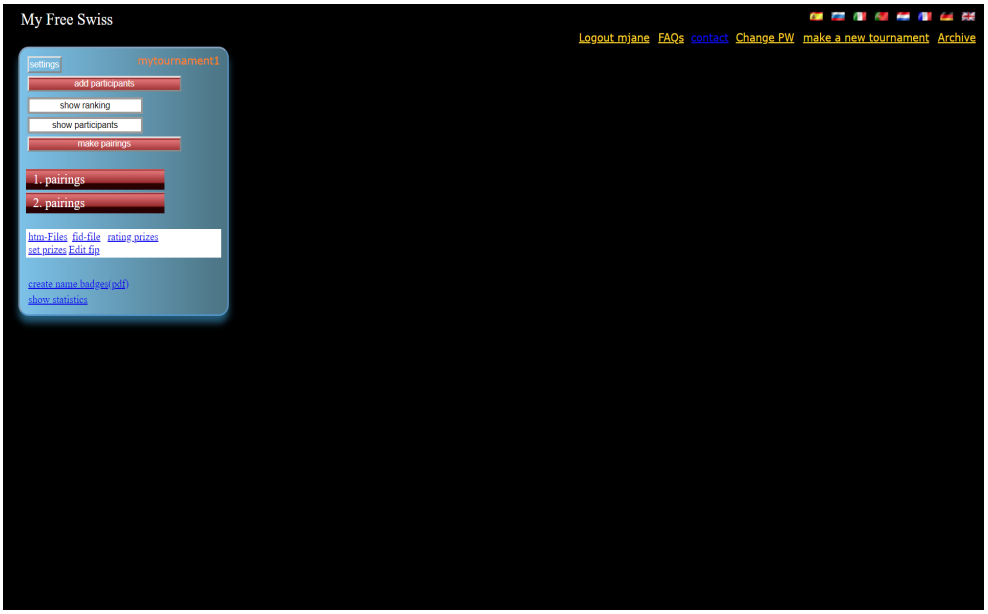
Obrázek 2.2: Vega – zadávání výsledků

Vega

Autor: Luigi Forlano (ITA)
Webová stránka: `www.vegachess.com`
Cena: Linux zdarma, Windows 50 €

Program Vega získal licenci FIDE v roce 2006. Nejnovější verze pak vznikla v roce 2012. [14] Obsahuje v podstatě stejnou funkcionalitu jako Swiss Manager, ale její realizace je nepraktická a ovládání je časově náročné. Program není vůbec snadné ovládat, pokud s tím uživatel nemá předchozí zkušenosti. Chybí také řádná zpětná vazba programu a srozumitelný popis některých tlačítek, viz obrázek 2.2.

Oficiálně je používán zélandskou, australskou a španělskou šachovou federací. Ve Španělsku je integrován s webovou stránkou `www.info64.org` [15], kde jsou zveřejňovány turnaje hrané ve Španělsku. Jedná se o placený program, má ale neplacenou verzi pro Linux. [14]



Obrázek 2.3: Free-swiss – stránka po vytvoření turnaje

2.1.2 Webové aplikace

S webovými aplikacemi na losování turnajů jsem neměla před průzkumem žádné zkušenosti. Pro vyhledávání jsem použila vyhledávač od Google, do kterého jsem zadala následující slovní spojení: *online chess pairing program*.

Aplikace, které níže analyzuji, jsem vyhledala jako doporučení na stránce `www.chess.stackexchange.com`, kde se uživatel snažil nalézt levný nebo neplacený program, viz [16].

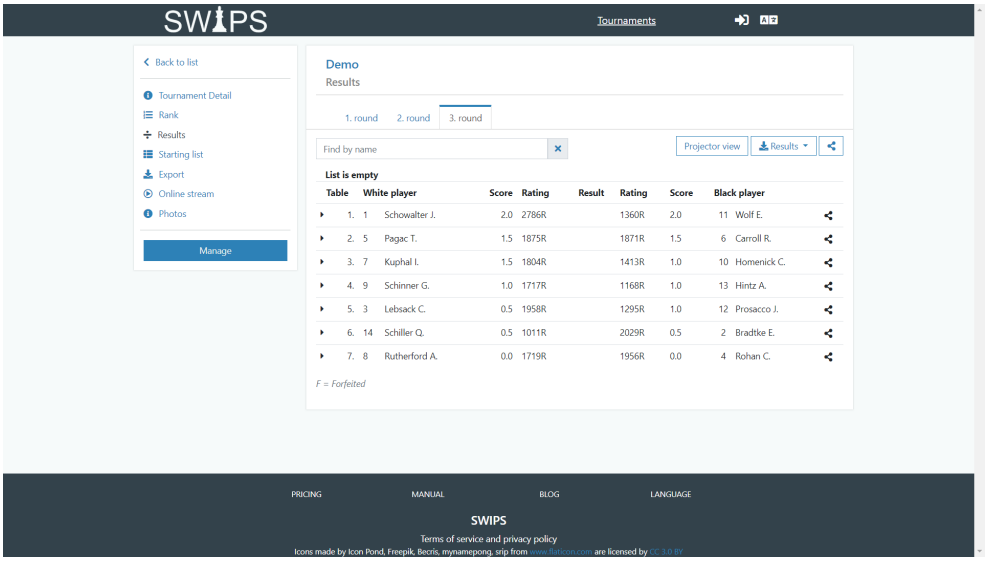
Free-Swiss

Web: `www.free-swiss.com`
Cena: zdarma

Tato webová stránka mi přišla velice špatná, co se týče grafického rozhraní. Na stránce se mísí velké množství barev různých odstínů a to vše na černém pozadí. Každá stránka má novou škálu barev a je velice nepřehledná. Hlavní stránka, po vytvoření turnaje, se skládá ze samých tlačítek nebo linků, viz obrázek 2.3, přičemž není úplně zřejmé, co jaké tlačítko dělá.

Přestože aplikace nabízela výběr z více jazyků, tak se po zvolení angličtiny na stránce mísila angličtina s němčinou.

Co se týče funkcionality, stránka nenabízí volbu pomocného hodnocení a nelze zadávat speciální výsledky. Zobrazuje pouze rozlosování kol a průběžná pořadí, ta se dají zobrazit v pdf formátu.



Obrázek 2.4: Swips – zobrazení kola

Swips

Web: `www.chess.swips.eu`
Cena: Light/Full verze: 5/10 €měsíčně

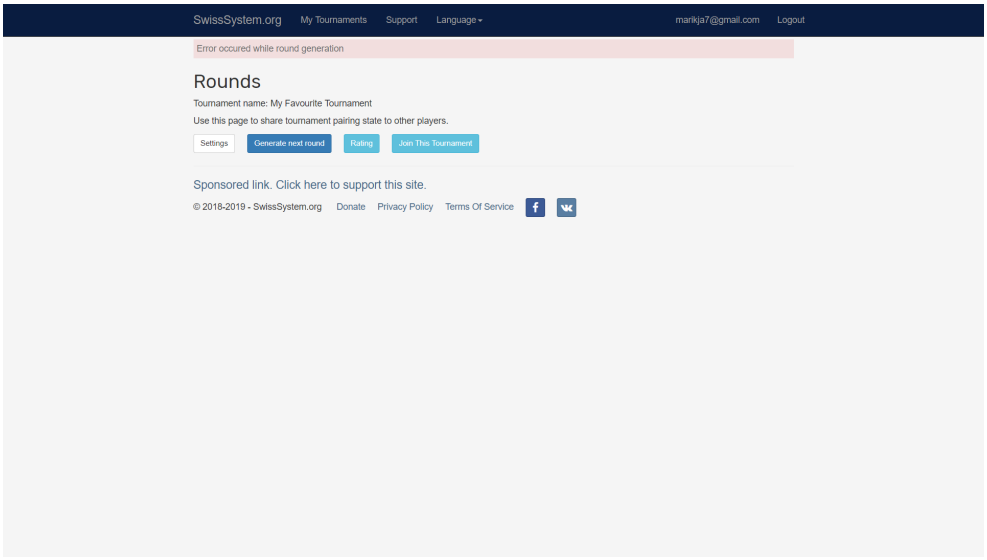
Mezi webovými aplikacemi jednoznačne vyčnívá Swips [17]. V současné době nemá licenci FIDE, ale pokouší se ji získat. Swips nabízí placenou a neplacenou verzi aplikace. Neplacená verze má striktní omezení na počet hráčů, počet kol a pomocné hodnocení. [18] Není proto vhodná pro reálné použití, ale pouze pro vyzkoušení práce s aplikací.

Narozdíl od ostatních zmiňovaných programů má poměrně moderní design a je uživatelsky přívětivá. Při používání programu jsem ale narazila na tlačítka, jejichž smysl mi nebyl úplně zřejmý. Na druhou stranu nabízí i funkčnosti, o kterých si myslím, že nejsou tak často využívané, jako je například import fotek z turnaje.

Při vytváření turnaje jsem se několikrát dostala do mrtvého bodu, kdy nebylo možné turnaj vytvořit a zároveň jsem nedostala žádné informace proč.

Aplikace nabízí možnost export souborů v rtf a png formátu. Tento export mi fungoval, pokud jsem zvolila možnost zobrazení vzorového turnaje. Pokud jsem ovšem chtěla exportovat soubory v reálném existujícím turnaji, daná tlačítka pro export tam nebyla.

Na žádné jiné chyby, které by měly negativní dopad na průběh turnaje, jsem nenarazila.



Obrázek 2.5: SwissSystem – chyba při generování kola

SwissSystem

Web: `www.swisssystem.org`
Cena: zdarma

Jedná se o prozatím nejnovější program na losování turnajů, konkrétně vznikl v roce 2018 [19]. Na tomto webu je zřejmé, že autor zvolil modernější technologie, patré je především použití bootstrapu na úpravu designu stránky. Stránka ale není dobře graficky zpracována a je využita pouze malá část obrazovky. To způsobuje, že se uživatel musí pracně proklikávat i pro využití základních funkčností. Navigace na stránce je členěna dost nelogicky. Ovládání většího turnaje by proto bylo časově velice náročné.

Chybí mu také některé základní funkcionality jako je výběr pomocného hodnocení a nastavení speciálních výsledků. Pomocné hodnocení je nastavené defaultně a uživatel nemá možnost si ho přizpůsobit.

Uživateli je dána poměrně velká volnost. Má například možnost zadat dva hráče se stejnými údaji, vkládat hráče do již zahájeného turnaje, mazat jednotlivá kola nebo měnit počet kol v průběhu turnaje.

Při práci s tímto programem jsem vytvořila přibližne deset turnajů, přičemž u jednoho nastala chyba při generováním kol, viz obrázek 2.5. Bohužel jsem nedokázala blíže identifikovat, čím byla způsobena. V turnaji bylo přihlášeno šest hráčů pro pět kol a systém turnaje byl švýcarský.

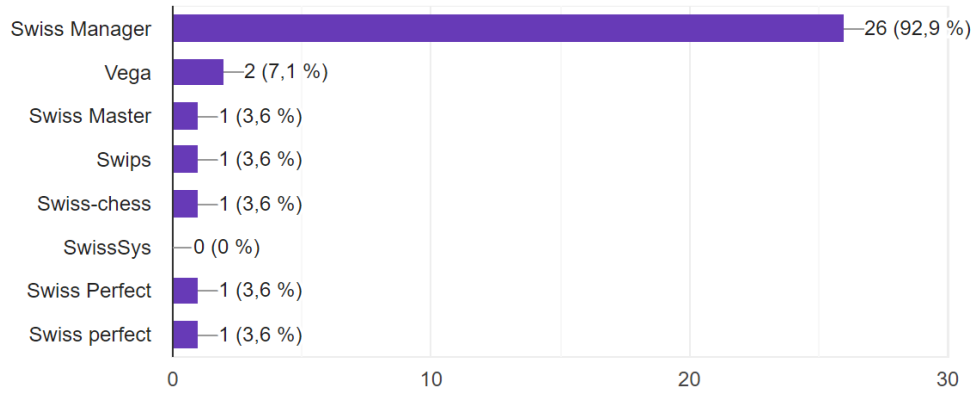
Kladně hodnotím to, že na domovské stránce aplikace je dostupné video, které práci s programem instruuje.

	Výhody	**Nevýhody**
Swiss Master	– licence FIDE – vlastní párovací program	– zastaralý design – placené – pouze na Windows – náročné používaní
Swiss Manager	– licence FIDE – mezinárodní použití – integrace s webem – licence koupená ŠSČR – integrována s webem	– zastaralý design – placené – pouze na Windows – náročné používaní
Vega	– licence FIDE – zdarma pro Linux – Dubov systém	– zastaralý design – placené pro Windows – náročné používání – nesrozumitelný popis tlačítek – integrována s webem pouze ve Španělsku
Swips	– moderní – uživatelsky přívětivé	– placené – u některých tlačítek chybí srozumitelný popis – nemá licenci FIDE
Swiss Sys	– zdarma – použití moderních technologií – instruktážní video	– nedostatečná funkcionalita – nepřehlednost – nelogická navigace
Free-Swiss	– zdarma	– velmi špatný design – nedostatečná funkcionalita – míchání angličtiny a němčiny

Tabulka 2.1: Shrnutí existujících programů

2.1.3 Srovnání programů

Většina ze zmíněných programů jsou zpoplatněny. Na internetu lze dohledat několik opensource programů, žádný z nich ale není dotažen do použitelné podoby. Všechny desktopové programy sloužící k losování turnajů si jsou velice podobné a jejich největší nevýhodou je zastaralý design a uživatelsky náročné ovládání. Mezi webovými aplikacemi jednoznačně vyčnívá aplikace Swips, ostatní aplikace mi nepřišly příliš povedené.



Obrázek 2.6: Používané programy

2.2 Cílová skupina uživatelů

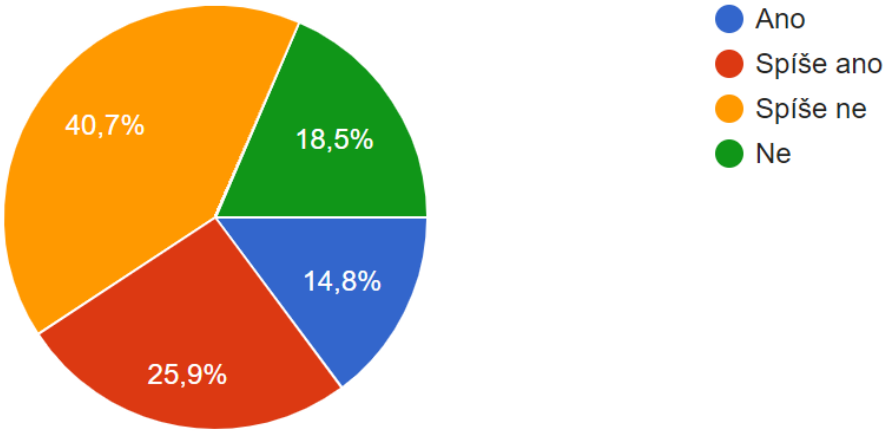
Před každým návrhem a implementací programu je vždy nejprve nutné zanalyzovat cílovou skupinu uživatelů. V mém případě obsahuje cílová skupina dvě složky – organizátory šachových turnajů a samotné hráče. Abych získala přehled o tom, které z analyzovaných programů se používají, vytvořila jsem dotazník pro orgnizátory turnajů, který se skládá z následujících otázek:

1. S jakými programy na losování turnajů máte zkušenosti?
2. S kterými z daných programů pracujete pravidelně?
3. Máte s daným programem kladné zkušenosti?
4. Přijde vám program uživatelsky přívětivý?
5. Sdílíte výsledky jednotlivých kol na internetu?

Tento dotazník jsem distribuovala mezi osoby, u kterých jsem věděla, že mají nebo měli 2. rozhodcovskou třídu. Tuto třídou může získat libovolná osoba starší 16. let po složení patřičných zkoušek. Jedním z bodů zkoušky je prověření znalosti práce se Swiss Managerem.

Na dotazník určený pro organizátory šachových turnajů zodpovědělo celkem 28 osob. Naprostá většina z nich znala a pravidelně používala pouze program Swiss Manager, viz graf 2.6. Pouze dva uživatelé pravidelně používali jiný program a to program Vega a program Swiss Perfect. Na otázku, zda mají s daným programem kladné zkušenosti, zodpovědělo 16 osob ne nebo spíše ne, viz graf 2.7. Práce s programem nepřijde intuitivní 21 osobám. Aktuální stav turnaje pak sdílí na internet 17 osob.

Co se týče samotných hráčů turnajů, tak ti jsou odkázání na to, co jim organizátoři nabídnou. To se odvíjí od toho, jaký program používají. Mnoho



Obrázek 2.7: Spokojenost s pravidelně používaným programem

turnajů je losováno pomocí Swiss Manageru u něhož organizátoři využívají propojení s chess-results.

Na chess-results si pak hráči mohou dohledat výsledky jednotlivých kol, průběžná pořádí a rozlosování aktuálního kola. Z mého pohledu je pak velice přínostné to, že si hráč může zobrazit profil libovolného hráče v turnaji. Na tom se dozví, s kým hrál, jakou měl barvu a výsledek zápasu.

2.3 Shrnutí

Z průzkumu existujících řešení vyplynulo, že neexistuje program, který by pokrýval veškerou potřebnou funkcionalitu a byl současně zdarma.

Analýzou cílové skupiny uživatelů jsem zjistila, že nejpoužívanějším programem mezi dotazovanými uživateli je Swiss Manager, ale mnoho uživatelů s ním není spokojeno a práce s programem jim nepřijde intuitivní. Důležitým zjištěním byl fakt, že velká část uživatelů sdílí aktuální stav turnaje online.

Z obou průzkumů vyplynulo, že nový program může v této oblasti přinést mnoho zlepšení a jeho tvorba není tedy zbytečná.

Kapitola 3

Návrh

V této kapitole nastíním požadavky na systém a rozeberu základní návrh a architekturu aplikace.

3.1 Specifikace požadavků

Na základě analýzy již existujících programů a vlastních zkušeností jsem sestavila seznam požadavků na funkcionalitu, kterou první nasazená verze aplikace bude obsahovat.

3.1.1 Funkční požadavky

Funkční požadavky specifikují očekávané chování programu. Jedná se o funkcionaliltu, která musí být implementována, aby uživatel mohl dosáhnout svých cílů. [20]

F1: Registrace a přihlášení

Aplikace bude umožňovat registraci a následné přihlášení již registrovaného uživatele. Uživatel bude mít také možnost používat aplikaci bez přihlášení. Pokud tak učiní, bude mít přístupnou omezenou funkcionalitu.

F2: Správa turnaje

Aplikace bude nabízet možnost vytvořit turnaj s potřebnými atributy:

1. Základní informace – název, datum, místo konání turnaje.
2. Systém – typ systému, pomocná hodnocení, bodové hodnocení.
3. Práva – zda bude turnaj veřejný.

Po vytvoření turnaje bude možné turnaj smazat.

F3: Editace turnaje

Mazání a úprava údajů o turnaji. Před začátkem turnaje (tedy před ukončením registrace), bude možné upravovat informace o systému turnaje. Ostatní informace se budou dát upravovat po celou dobu turnaje.

F4: Zobrazení turnajů

Aplikace bude zobrazovat všechny dostupné turnaje pro daného uživatele. Pro snadnější nalezení daného turnaje bude možná filtrace a vyhledávání turnajů.

F5: Zobrazení detailu turnaje

Aplikace umožní zobrazení všech potřebných informací o turnaji jako jsou základní informace, průběžné pořadí, rozlosování a historie kol.

F6: Evidence hráčů

Před zahájením turnaje bude možné přidávat a odebírat hráče. Hráče bude možné přidávat z existující databáze reálných hráčů.

F7: Losování kol

Aplikace bude umožňovat losování kol. Kola se budou losovat v závislosti na volbě systému při vytvoření turnaje či jeho editaci.

1. **Švýcarský systém** – jednotlivá kola turnaje se budou losovat postupně. Dané kolo se bude dát losovat, pokud bylo ukončeno předchozí kolo.

2. **Round-robin** – všechna kola turnaje se nalosují najednou. Zadávání výsledků bude možné pouze u aktuálního kola.

Všechna nalosovaná kola budou uživatelům přístupná.

F8: Zadávání výsledků

Kromě klasických výsledků (1–0, 0–1, ½–½) bude program umožňovat zadávat i speciální výsledky (1F–0F, 0F–1F, ½F–½F).

F9: Zobrazení profilu hráče

Aplikace bude umožňovat zobrazení profilu libovolného hráče v tunaji. Na tomto profilu budou základní informace o hráči (jméno, elo, titul) a s kým jednotlivá kola hrál, jakou měl barvu a výsledek zápasu.

F10: Export tabulek

Aplikace bude umožňovat export důležitých listin a tabulek v CSV formátu. CSV formát je standartizovaný formát pro ukládání tabulkových dat [21]. Exportovat se budou dát průběžná pořádí, startovní listina a jednotlivá kola.

3.1.2 Nefunkční požadavky

Nefunkční požadavky hrají klíčovou roli při vývoji programu. Na jejich základě se rozhoduje o zvolené architektuře, výběru technologií, platformě atd. [22]

NF1: Nezávislost na platformě

Z nalezených programů, které obsahují potřebnou funkcionalitu, byly pouze dva z nich spustitelné na Linuxu. O spustitelnosti na macOS většinou nebyla ani zmínka. Proto je nezávislot na platformě klíčová, pro reálné prosazení programu.

NF2: Využití moderních technologií

Jedná se o program, který bude dostupný jako opensorce. Pro případ, že by program chtěl někdo rozšířit je důležité použít technologie, které jsou často používané.

NF3: Uživatelská přívětivost

Jeden ze stěžejních bodů je intuintivní ovládání a logicky rozvržená funkcionalita programu.

NF4: Dostupnost

Aplikace musí být nasazená a dostupná alespoň pro uzavřenou skupinu uživatelů.

NF5: Rozšiřitelnost

Aplikace musí být navrhnuta s důrazem na možné další rozšíření aplikace. Proto musí být zvolená jednotná struktura a architektura aplikace. Za tímto účelem bude vytvořena i dokumentace.

3.2 Role

V návrhu aplikace rozlišuji tři typy rolí:

- **Nepřihlášený uživatel** – uživatel, který navštíví stránku bez přihlášení, tedy jako *host*.
- **Přihlášený uživatel** – registrovaný uživatel, který se přihlásí pomocí svého uživatelského jména a hesla.
- **Administrátor turnaje** – uživatel, který má přidělené administrátorké oprávnění nebo turnaj vytvořil. Administrátor turnaje bude mít vždy současně roli přihlášeného uživatele.

Aktér je pak uživatel, který používá systém a zastává určitou roli.

3.3 Případy užití

Případy užití jsou souhrn možností jak používat systém tak, aby konkrétní uživatel dosáhl svých cílů [23]. Jejich účelem je definovat rozsah systému a identifikovat, co se během dané interakce mezi uživatelem a systémem může pokazit [24].

Případy užití by měly dle [25] zahrnovat:

- Kdo aplikaci používá (aktér).
- Co chce uživatel udělat.
- Jak toho dosáhne.
- Co tím získal.
- Jak by měla stránka reagovat.

V programu identifikuji následující případy užití:

UC1: Registrace
Pokud uživatel ještě není na stránce registrovaný má možnost vytvořit účet po odeslání uživatelského jména a hesla.

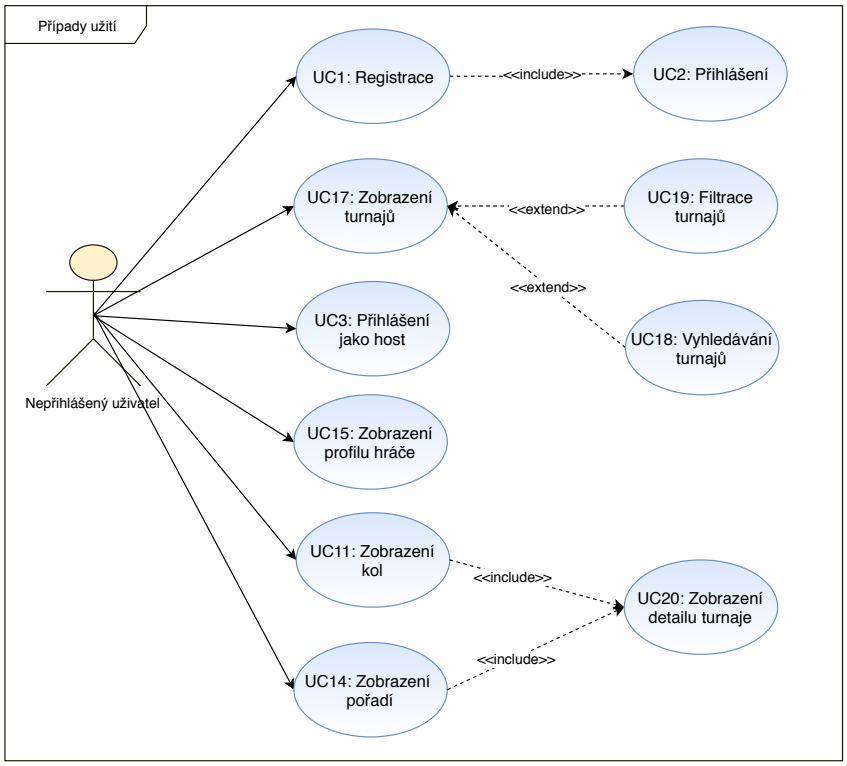
1. Uživatel vyplní své uživatelské jméno a heslo do formuláře *Register*.
2. Uživatel odešle formulář.
3. Systém uživatele přesměruje na stránku s výpisem turnajů.

UC2: Přihlášení
Pokud je uživatel na stránce registrovaný má možnost se přihlásit po odeslání uživatelského jména a hesla.

1. Uživatel vyplní své uživatelské jméno a heslo do formuláře *Login*.
2. Uživatel odešle formulář.
3. Systém uživatele přesměruje na stránku s výpisem turnajů.

UC3: Přihlášení jako host
Uživatel, který nemá zájem se registrovat, má možnost navštívit stránku bez registrace a přihlášení. Nebude mít ale stejná práva jako přihlášený uživatel.

1. Uživatel klikne na tlačítko *Login as a guest*.
2. Systém uživatele přesměruje na stránku s výpisem turnajů.



Obrázek 3.1: Diagram případů užití

UC4: Vytvoření turnaje

Pokud je uživatel přihlášený má možnost vytvořit turnaj.

1. Navigace na domovskou stránku.
2. Uživatel klikne na tlačítko *New Tournament*.
3. Systém uživatele přesměruje na stránku s formulářem.
4. Uživatel vyplní informace a klikne na tlačítko *Next*.
5. Systém uživateli zobrazí formulář se systémem turnaje.
6. Uživatel vyplní informace a klikne na tlačítko *Next*.
7. Systém uživateli zobrazí formulář s právami turnaje.
8. Uživatel vyplní informace a klikne na tlačítko *Create*.
9. Systém uživatele přesměruje na domovskou stránku turnaje.

UC5: Editace údajů turnaje
Uživatel má možnost editovat informace o turnaji po vytvoření turnaje.

1. Navigace na stránku *Settings*.
2. Uživatel klikne na inkonu *edit*.
3. Systém zobrazí formulář v editačním modu.
4. Uživatel vyplní nové údaje a odešle je.

UC6: Přidání hráčů do turnaje
Po vytvoření turnaje, je možnost přidávat hráče do turnaje.

1. Navigace na stránku *Registration*.
2. Uživatel vyplní potřebné údaje do formuláře.
3. Přidat hráče s danými údaji pomocí tlačítka plus.
4. Systém uživateli zobrazí seznam přidaných hráčů.

Alternativní scénář:

1. Navigace na stránku *Registration*.
2. Uživatel zapne vyhledávání v databázi reálných hráču.
3. Uživatel zadá údaje o hráči do formuláře.
4. Systém uživateli zobrazí všechny hráče se shodnými údaji.
5. Uživatel přidá reálného hráče z databáze do turnaje.

UC7: Odstranění hráče z turnaje
Před zahájením turnaje má uživatel možnost odstranit hráče z turnaje.

1. Navigace na stránku *Registration*.
2. Uživatel klikne na tlačítko *delete* u hráče, kterého chce odstranit.
3. Systém smaže daného hráče z turnaje.

UC8: Zahájení turnaje
Po přidání dostatečného počtu hráčů má uživatel možnost zahájit turnaj.

1. Navigace na stránku *Registration*.
2. Uživatel klikne na tlačítko *End registration*.

UC9: Losování kol švýcarským systémem

Po zahájení turnaje (ukončení registrace) je možné postupně losovat jednotlivá kola švýcarského systému.

1. Navigace na stránku *Pairings*.
2. Uživatel klikne na tlačítko *Loss Round*.
3. Systém uživatele přesměruje na práve nalosované kolo.

UC10: Losování kol round-robin systémem

Po zahájení turnaje s nastavením round-robin se vylosují všechna kola.

1. Navigace na stránku s registrovanými hráči.
2. Uživatel klikne na tlačítko *End Registration*.

UC11: Zobrazení kol

Uživatel má možnost zobrazit si všechna zveřejněná kola turnaje (tedy ta, která již byla nalosována).

1. Navigace na stránku *Rounds*.
2. Uživatel klikne na kolo, které si chce zobrazit.
3. Systém uživatele přesměruje na dané kolo.

UC12: Zadávání výsledků

Po rozlosování kol je možnost zadávat výsledky k aktuálnímu kolu turnaje.

1. Navigace na stránku s rozlosováním aktuálního kola.
2. Uživatel rozklikne možné výsledky u dané partie.
3. Uživatel vybere z možných výsledků.
4. Systém zobrazí vybraný výsledek u dané partie.

UC13: Ukončení kola

Pokud jsou zadány všechny výsledky je možné ukončit kolo. Po ukončení kola již není možné upravovat výsledky.

1. Navigace na stránku s rozlosováním aktuálního kola.
2. Uživatel klikne na tlačítko *End Round*.
3. Systém zobrazí dané kolo, bez možnosti výběru výsledků.

UC14: Zobrazení pořadí

Uživatel má k dispozici průběžné pořadí po všech ukončených kolech.

1. Navigace na detail turnaje.
2. Uživatel si pomocí menu turnaje zobrazí průběžné pořadí po jednotlivých kolech.
3. Systém uživateli zobrazí všechna kola, která již byla odehrána.
4. Uživatel si vybere kolo, po kterém chce zobrazit průběžné pořadí.
5. Systém uživateli zobrazí průběžné pořadí po daném kole.

UC15: Zobrazení profilu hráče

Uživatel má možnost podívat se na profil každého hráče v turnaji.

1. Navigace na startovní listinu hráčů.
2. Uživatel klikne na daného hráče na startovní listině.
3. Systém uživatele přesměruje na stránku s profilem hráče.

UC16: Export tabulek

Uživatel má možnost stáhnout průběžná hodnocení, rozlosování kol a startovní listinu.

1. Navigace na stránku s tabulkou, kterou chce exportovat.
2. U tabulky klikne na tlačítko *Export*.
3. Systém uživateli tabulku stáhne ve formátu CVS.

UC17: Zobrazení turnajů

Na hlavní stránce turnaje budou zobrazené všechny dostupné turnaje.

1. Navigace na domovskou stránku.
2. Systém vrátí všechny turnaje, ke kterým má uživatel přístup.

UC18: Vyhledávání turnajů

Uživatel má k dispozici search bar, pomocí kterého může vyhledávat turnaje.

1. Navigace na domovskou stránku turnaje.
2. Uživatel zadá část názvu turnaje do search baru.
3. Systém vrátí turnaje, které obsahují daný podřetězec.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
UC1	✓									
UC2	✓									
UC3	✓									
UC4		✓								
UC5			✓							
UC6						✓				
UC7						✓				
UC8								✓		
UC9								✓		
UC10								✓		
UC11					✓					
UC12								✓		
UC13							✓			
UC14					✓					
UC15									✓	
UC16										✓
UC17					✓					
UC18					✓					
UC19				✓						

Tabulka 3.1: Pokrytí funkčních požadavků

UC19: Filtrace turnajů

Uživatel má k dispozici filtr turnajů podle práv.

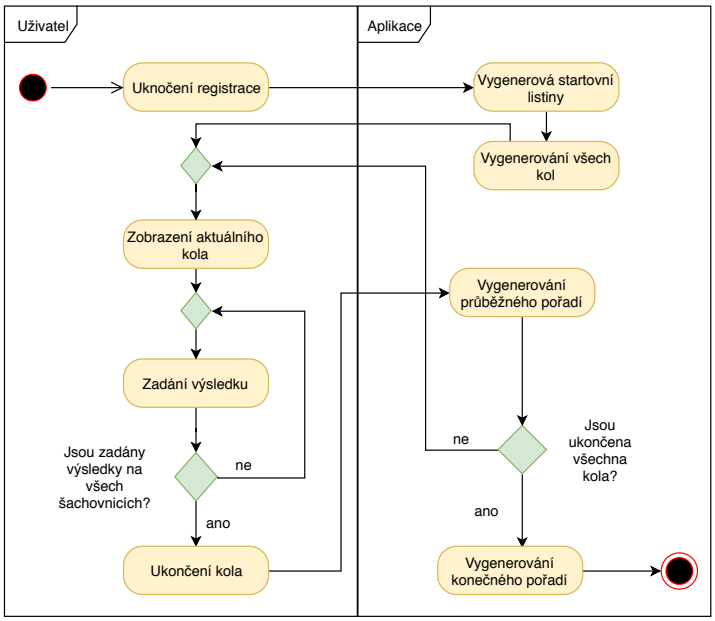
1. Uživatel rozklikne možnosti filtrace a vybere odpovídající možnost.
2. Systém vrátí turnaje s odpovídajícími právy.

UC20: Zobrazení detailu turnaje

Uživatel si může zobrazit detail turnaje.

1. Navigace na domovskou stránku.
2. Uživatel klikne na daný turnaj.
3. Systém uživatele přesměruje na domovskou stránku turnaje.

Případy užití jsou pouze rozšířením funkčních požadavků o možnosti uživatele, proto by se měly plně pokrývat. Tabulka 3.1 toto pokrytí znázorňuje.



Obrázek 3.2: Průběh turnaje round-robin systémem

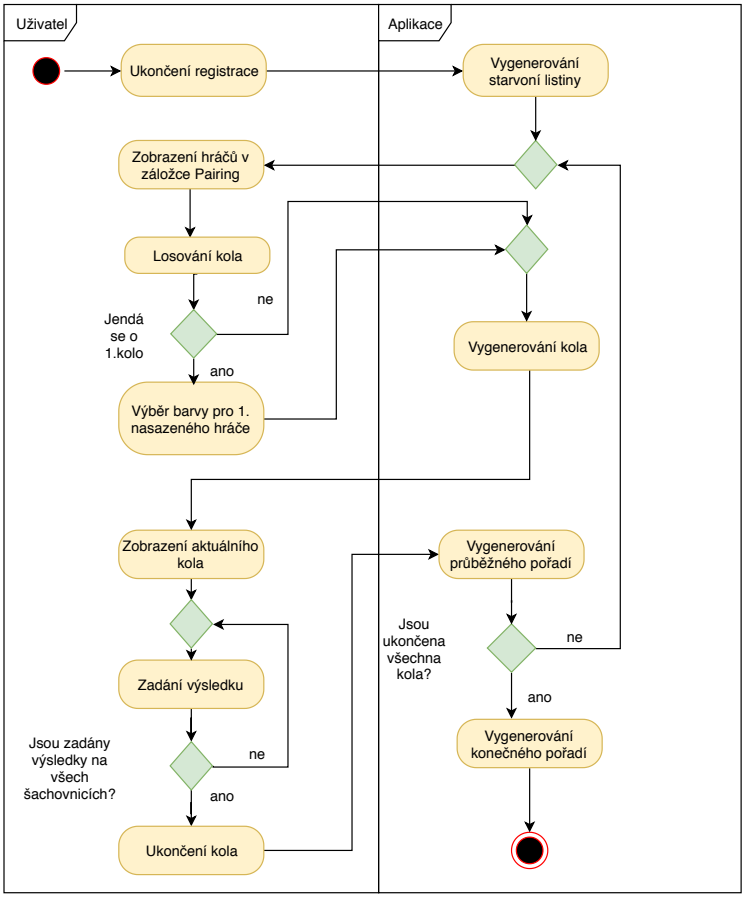
3.4 Diagram aktivit

Diagram aktivit je jedním z UML diagramů. Graficky znázorňuje činnost systému, tok dat a možnosti rozhodování. Používá se například pro modelování případů užití. [26]

Průběh turnaje systému round-robin se v mnoha ohledech liší od průběhu turnaje švýcarským systémem. Proto popíšu a pomocí diagramu aktivit namodeluji obě možnosti průběhu turnaje, aby byl daný rozdíl zřejmý. Průběh turnaje round-robin je namodelován na obrázku 3.2 a průběh turnaje šýcarským systémem je zobrazen na obrázku 3.3.

3.4.1 Průběh turnaje round-robin systémem

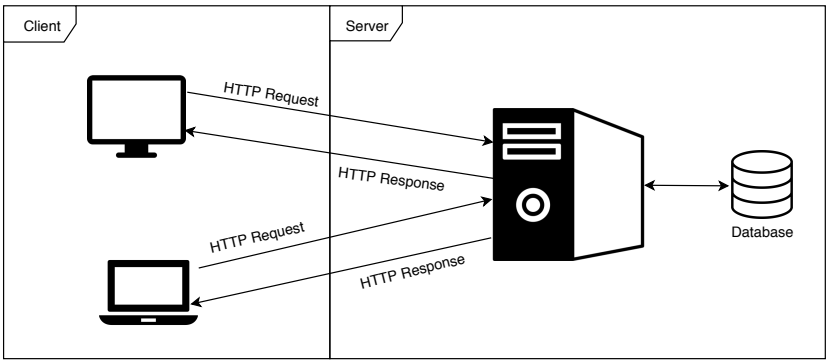
Nacházíme se v okamžiku, kdy uživatel úspěšně vytvořil turnaj se systémem round-robin a do turnaje jsou již přidáni všichni požadovaní hráči. Nyní má uživatel možnost ukončit registraci. Po ukončení registrace se vygeneruje startovní listina hráčů a všechna kola turnaje. Následně má oprávněný uživatel možnost zadávat výsledky k aktuálnímu kolu. Pokud uživatel zadal výsledky k danému kolu, má možnost ho ukončit. Po ukončení kola se vždy vygeneruje nové průběžné pořadí po daném kole. Po zadání výsledků k poslednímu vygenerovanému kolu, se vygeneruje konečné pořadí hráčů.



Obrázek 3.3: Průběh turnaje švýcarským systémem

3.4.2 Průběh turnaje švýcarským systémem

Nacházíme se v okamžiku, kdy uživatel úspěšně vytvořil turnaj s švýcarským systémem a do turnaje jsou již přidáni všichni požadovaní hráči. Nyní má uživatel možnost ukončit registraci. Po ukončení registrace se vygeneruje startovní listina hráčů. Oprávněný uživatel má možnost navigovat se na stránku s registrovanými hráči v turnaji. Zde se nachází tlačítko pro nalosování kola. Po nalosování kola se dané kolo vygeneruje. Následně má oprávněný uživatel možnost zadávat výsledky k aktuálnímu kolu. Pokud uživatel zadal výsledky k danému kolu, má možnost ho ukončit. Po ukončení kola se vždy vygeneruje nové průběžné pořadí pro dané kolo. Po zadání výsledků k poslednímu vygenerovanému kolu, se vygeneruje konečné pořadí hráčů.



Obrázek 3.4: Architektura klient-server [27, 28]

3.5 Základní architektura

V momentě, kdy jsou vyjasněny všechny požadavky na systém, je nutné rozhodnout na jaké platformě aplikace poběží a jaká bude její základní architektura.

Z funkčních a nefunkčních požadavků je zřejmé, že se program musí realizovat jako webová aplikace, nebo desktopová aplikace, která by ale měla přístup na internet.

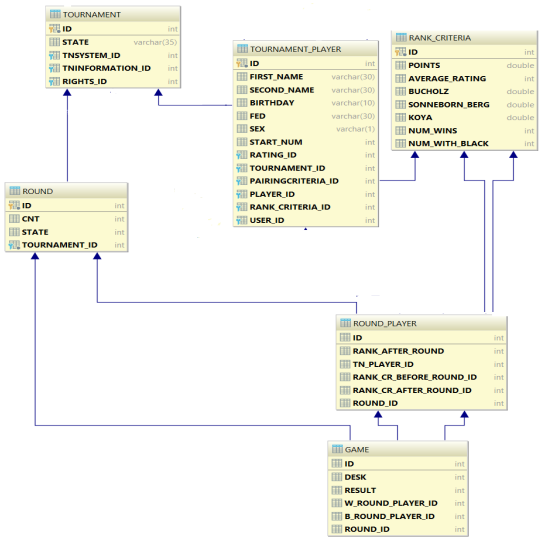
Vycházela jsem z předpokladu, že ve většině hracích místností je nutný přístup na internet a proto jsem zvolila čistě webovou aplikaci. Výhodou webových aplikací je, že fungují nezávisle na operačním systému a také to, že organizátoři nebudou muset řešit sdílení aktuálního stavu turnaje mezi hráče a jednoduše nastaví, jestli bude turnaj veřejný nebo soukromý.

Mou motivací k tomuto rozhodnutí byl i fakt, že existují framworky, které umožňují webovou aplikaci zabalit do desktopové aplikace. Proto budu nad tvorbou desktopové aplikace uvažovat, jako o možnosti rozšíření, viz kapitola 7.3.

Hlavní návrh aplikace je založen na architektuře klient–server. Tato architektura je znázorněna na obrázku 3.4. Výhodou této architektury je, že klient a server se můžou vyvíjet nezávisle a server může komunikovat s více klienty. Tedy desktopová aplikace bude moci využívat stejný server.

3.6 Databázový model

Důležitou součástí aplikace je databáze. Kvalita návrhu databáze hraje klíčovou roli v celkové efektivitě aplikace. Databázový model popisuje logickou strukturu relační databáze. Důvod pro zvolení relační databáze je popsán v kapitole 4.4. Na obrázku 3.5 je znázorněna pouze její část. Model celé databáze je umístěn do přílohy C. Níže je pak popsán význam některých tabulek.



Obrázek 3.5: Část databázového modelu

- **`Tournament`** – obsahuje údaje o turnaji. Je propojen s entitami `TnSystem` a `TnInformation`. Tyto entity obsahují doplňující informace. `TnSystem` upřesňuje systém turnaje a `TnInformation` obsahuje základní informace o turnaji.
- **`SecureUser`** – slouží k uložení dat uživatele po registraci. Obsahuje vazbu s entitami `Tournament`, které daný uživatel vytvořil.
- **`Player`** – na stránkách FIDE je umožněno stažení XML dokumentu, který obsahuje reálné hráče. Tato tabulka je naplněna naparsovanými daty právě z tohoto souboru.
- **`TournamentPlayer`** – tato tabulka musí být vždy propojena s právě jedním turnajem. Představuje jednu instanci `Player`, který se účastní daného turnaje.
- **`Round`** – představuje jedno kolo turnaje.
- **`RoundPlayer`** – představuje instanci hráče pro jedno kolo. V aplikaci často vypisuji hráče pro určité kolo a proto je zbytečné vypočítavat vždy body a pomocná hodnocení. Každý `RoundPlayer` je namapován právě na jednoho `TournamentPlayer`.
- **`Game`** – představuje instanci hry v kole. Každé kolo (`Round`) má n her.

Kapitola 4

Použité technologie

V následující kapitole představím technologii, kterou jsem zvolila k implementaci aplikace a motivaci k jejímu výběru.

4.1 Klient

Klient představuje tu část aplikace, která má za úkol přímou komunikaci s uživatelem. Na základě požadavků uživatele posílá dotazy na server, se kterým komunikuje pomocí REST, viz kapitola 4.3.1.

Na tvorbu klienta jsem využila CSS, HTML a JavaScript. HTML definuje strukturu a obsah webových stránek [29], CSS pak pomáhá přizpůsobit jejich vzhled. JavaScript je znám především jako scriptovací jazyk používaný pro webové stránky. [29] Vznikl v roce 1995 a od té prošel mnoha změnami, jak v možnostech použití, tak v syntaxi. Aktuálně se používají především dvě syntaxe JavaScriptu ES5 a ES6. ES6 je rozšířením ES5 a zavádí nové syntaktické prvky, které pomáhají zpřehlednit kód. Ne všechny prohlížeče podporují ES6, proto je někdy potřeba překlad ES6 na ES5, který zproztředkovává nástroj Babel. [30] Jako nadstavba nad JavaScriptem je vytvořen TypeScript, který se na JavaScript překládá.

Při výběru technologie na implementaci frontendu jsem se rozhodovala mezi frameworkem Angular a knihovnou React.

4.1.1 Angular

Nad Angularem jsem uvažovala především proto, že jsem s ním měla předchozí zkušenosti. Angular je framework založen na TypeScriptu a je objektově orientován. Existuje více verzí Angularu a všechny se od sebe dost liší. V podstatě se jedná o úplně jiné frameworky a tak nastává problém s kompatibilitou. Tento způsob vývoje se mi úplně nezamlouvá. Proto jsem se rozhodla seznámit s novou technologií a zvolila jsem React. Mé rozhodnutí podpořil i fakt, že se podle [31] jedná o nejpoužívanější technologii na frontend.

4.1.2 React

React je JavaScriptová knihovna. Jeho hlavní stavební cihlou jsou komponenty. Komponenta implementuje funkci `render()`, ta vrací HTML, které se vykreslí na obrazovce. Mimo jiné obsahuje i JavaScipt kód. Tedy HTML není odděleno od samotné logiky, jako je tomu například v Angular. Komponenta také obsahuje objekt `state`, do kterého lze uložit data. Po přepsání state se zavolá funkce `render()`, která vykreslí aktuální obsah komponenty. Většina ostatních frameworků je založena na architektuře *model–view–controller* neboli MVC. React v tomto modelu představuje pouze *view*. To znamená, že popisuje uživatelský interface a způsob jak se daný interface mění v závislosti na datech. [32]

4.1.3 Redux

Pro lepší manipulaci s načtenými daty jsem využila knihovnu Redux. Díky této knihovně je možnost do definovaných stavů uložit načtená data. Tato data jsou uložena ve `store`. Reducer mění data v závislosti na action, jenž má za úkol rešit logickou část a komunikaci se serverem. V action se řeší validace získaných dat a směrování na reducer pomocí `type`.

Tyto stavy lze předávat mezi jednotlivými komponentami a není tedy nutnost po každém routováním načítat data znovu.

Stav je jednoduchý objekt se stromovou strukturou. Lze měnit pouze v action, v ostatních částech klienta je přístupný pouze ke čtení.

Podle [33] Redux funguje na třech základních principech:

1. **Jediný zdroj pravdy** – Redux se skládá z jediného `store`, který obsahuje celkový stav alikace.

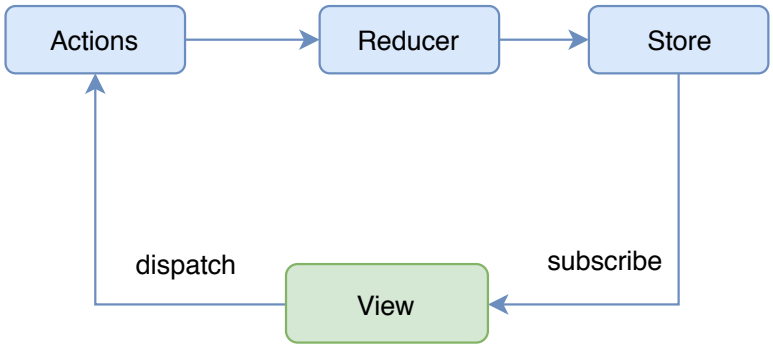
2. **Stavy pouze ke čtení** – stavy lze měnit pouze pomocí `dispatch action`.

3. **Změny stavu řízeny jednoduchými funkcemi** – pokud Reducer dostane stejný `type`, vrátí vždy stejný `state`.

4.1.4 Bootstrap

Bootstrap je framework pro vývoj webových a mobilních aplikací. Obsahuje designové templety založené na HTML a CSS a je podporován ve všech moderních prohlížečích. Při použití Reactu lze použít react-bootstrap, který reimplementuje JavaScript klasického bootstrapu. [34]

Stejnou funkcionalitu nabízí například material design od společnosti Google nebo Material Design for Bootstrap, který má ale některé temlaty placené.



Obrázek 4.1: Architektura klienta

4.2 Server

Server řeší hlavní logiku aplikace. Má za úkol reagovat na požadavky klienta a v reakci na to načítat, ukládat nebo mazat potřebná data z databáze.

Aktuálně je podle [35] nejpoužívanějším jazykem na backend jednoznačně Java. V dnešní době se používají především dva frameworky, založené na Jave, které slouží k vývoji backendu, Java EE a Spring. K realizaci serverové části aplikace jsem zvolila Spring. K rozhodnutí mě vedlo za prvé to, že je aktuálně nejrozšířenějším frameworkem na backend založen na Jave a za druhé jsem s ním měla předchozí kladné zkušenosti.

4.2.1 Spring

Spring vznikl v roce 2003 jako reakce na komplexnost specifikací Java EE. Mnoho těchto specifikací je tedy z Java EE převzato. Později vznikl Spring Boot, který ze Spring vychází a umožňuje spustit projekt aniž by uživatel musel zdlouhavě nastavovat jeho konfiguraci. Obsahuje například Tomcat server, aplikaci tedy lze jednoduše spouštět jako JAR soubor. Spring Framework je rozdělen do modulů. Záleží na aplikaci, které moduly využije.

4.3 Komunikace klient–server

Komunikace mezi servrem a klientem probíhá tak, že server poslouchá na určitém portu a klient pak na daný port odesílá data. Tato komunikace vychází z protokolu HTTP.

HTTP metoda	CRUD operace	význam
POST	Create	vytvoření
GET	Read	čtení dat
PUT/PATCH	Update	update dat
DELETE	Delete	smazání dat

Tabulka 4.1: HTTP operace

4.3.1 REST

Representational State Transferm je architektonický princip pro přístup ke zdrojům, který vychází z protokolu HTTP a používá se především pro vývoj webových aplikací. Zdroje mohou být data, stejně jako stavy aplikace. Všechny zdroje mají vlastní identifikátor Uniform Resource Identifier (URI). Další z principů REST je, že je bezstavový. [36]

REST specifikuje čtyři metody pro přístup ke zdrojům a poskytuje tak všchny operace CRUD, viz tabulka 4.1.

4.3.2 JSON

JavaScript Object Notation je formát ukládání dat. Je podporován většinou jazyků, proto se často tento formát používá na přenos dat mezi klientem a serverem. Níže je uveden zápis dat v JSON formátu.

```
[
    {
        "fideId": 334138,
        "firstName": "Marikova",
        "secondName": "Jana",
        "fed": "CZE",
        "sex": "F",
        "birthday": "1996",
        "rating": {
            "id": 394387,
            "elo": 2123,
            "title": "-"
        }
    },
    ...
]
```

Zdrojový kód 4.1: Ukázka formátu JSON

4.4 Databáze

Při výběru databáze je prvním rozhodnutím zda se má jednat o relační nebo nonrelační databázi.

Relační databáze využívá jako svůj základ koncept tabulek, které můžou odkazovat na jiné tabulky pomocí jejich id. Tabulky obsahují atributy (sloupce) a jednotlivé záznamy (řádky). Celkový model složený z tabulek musí být vytvořen před vkládáním dat. Na vložená data se lze dotazovat pomocí jazyku SQL. [37]

Jazyk SQL (Structured Query Language) je klasický jazyk pro operace s relační databází.

Nonrelační databáze dává uživateli daleko větší volnost. Využívá dynamické schéma pro strukturování dat. Existuje několik druhů nonrelační databáze například Klíč-hodnota, což je v podstatě asociativní pole, dokumentová databáze a další. [37]

Před tvorbou této práce jsem měla předchozí zkušenosti s nonrelační databází Firebase a relační databází Apache Derby. Vzhledem k tomu, že mi přišel koncept relační databáze lépe uchopitelný a vhodnější pokud je potřeba řešit transakce napříč daty, rozhodla jsem se opět zvolit Apache Derby.

Apache Derby je relační opensource databáze napsaná v Jave. Tato databáze poběží v libovolné Java Virtual Machine, je vyžadováno JDK. Poskytuje dvě možnosti nasazení:

- Serverové – určena, pokud má k databázi přístup více uživatelů napříč sítí. Běží v JVM, kterou poskytuje hostitelský server.
- Embededd – slouží pro použití pouze jedním klientem. Derby běží ve stejné virtual machine jako aplikace

Kapitola 5

Implementace

V této kapitole uvedu některé z důležitých částí implementace, které byly zajímavé z hlediska návrhu nebo realizace. Na závěr kapitoly představím interface výsledné aplikace.

5.1 Přístup k databázi

K přístupu k datům z databáze využívám Hibernate, která je implementována podle specifikace JPA, jež zprostředkovává modul ORM a JDBC. Tato specifikace pochází z Javy EE. Spring vytvořil nadstavbu nad JPA zvanou Spring Data, která ulehčuje práci s JPA.

JPA specifikuje chování násobnosti vztahů, typ fetchování a základní operace s databází, jako je mazání, vložení a další. U každého vztahu je možné definovat kaskádové typy a typ fetchování.

Typy fetchování jsou dva: `eager` a `lazy`. `Eager` natáhne všechna data i napříč tabulkami, kde existuje vazba. `Lazy` natáhne právě tu danou tabulku. Vzhledem k tomu, že při větším počtu uložených turnajů, bylo načítání metodou `eager` velmi pomalé, zvolila jsem ve většině entitách načítání typu `lazy`.

JPA je mocný nástroj pokud se jedná o aplikaci s menším počtem entit. Výhodou je, že implementuje základní funkce pro práci s databází. Problém nastává, když je potřeba psaní vlastních složitějších SQL dotazů. JPA má vlastní jazyk na psaní query dotazů, který podporuje SQL. Jednotlive SQL dotazy se definují v repository pomocí anotace `@Query`. Parametr anotace je pak vlastní SQL dotaz jako string.

Ve své práci jsem použila SQL dotazy především na výpočet pomocného hodnocení. Bez použití SQL dotazů by se muselo získávat zbytečné množství dat.

5.2 Registrace hráčů

Pokud by program vyžadoval ruční zadávání všech údajů, které potřebují organizátoři u hráče evidovat, byla by registrace velice zdlouhavý proces a bylo by pravděpodobné, že se u některého z hráčů zadá špatný údaj, což může mít negativní vliv na celý turnaj. Proto bylo nutné daný proces co možná nejvíce zjednodušit.

Pokud hraje šachový hráč na oficiálních turnajích, které se započítávají na FIDE elo, musí být registrován u organizace FIDE.

Na oficiálních stránkách FIDE se každý měsíc zveřejňuje aktuální FIDE listina se všemi registrovanými hráči v XML a TXT formátu. Tato listina je dostupná vždy na stejné url stránce `ratings.fide.com/download/players_list_xml.zip` pomocí metody `GET`. FIDE u každého hráče eviduje vždy stejné atributy, které jsou definovány pomocí dvojice tagů. Daný soubor jsem parsovala stavovým automatem sestrojeného pomocí konstruktu `Swich-case`. Příklad hráče v XML formátu vyjmutého z oficiální listiny je ukázán na zdrojovém kódu 5.1.

Uživatel má tedy možnost při přidávání hráčů do turnaje hledat hráče v této databázi podle jména a přijmení. V databázi se vždy vyhledává shoda s daným prefixem. Protože se jedná o databázi, která má okolo 700 000 záznamů, vracím pouze 10 odpovídajících výsledků. Stránka s registrací hráčů je na obrázku 5.2.

```
<playerslist>
  ...
  <player>
    <fideid>334138</fideid>
    <name>Marikova, Jana</name>
    <country>CZE</country>
    <sex>F</sex>
      ...
    <rating>2123</rating>
      ...
    <birthday>1996</birthday>
      ...
  </player>
  ...
</playerslist>
```

Zdrojový kód 5.1: Ukázka hráče ve formátu XML

pozice	**popis**	**obsah**
1–3	id	001
5–8	startovní číslo	od 1 do 9999
10	pohlaví	m/w
11–13	titul	GM, IM, WGM ...
15–47	jméno	přijmení, jméno
49–52	elo	
54–56	federace	
58–68	FIDE id	
70–79	datum narození	RRRR/MM/DD
81–84	body	
86–89	pořádí	

Pro každé kolo k:

92–95 $+(k-1)*10$	id oponenta	
$97+(k-1)*10$	barva	w/b/–
$99+(k-1)*10$	výsledek	

Tabulka 5.1: Příklad TRF formátu pro hráče

5.3 Holandský systém

V pruběhu implementace algoritmu holandského systému jsem narazila na program, který daný systém losuje. Tento losovací program používají všechny programy, které mají licenci FIDE, kromě Swiss Master, který má implementovaný vlastní losovací program. [10] Proto jsem se ho prozatím rozhodla použít a v další verzi aplikace dodělat vlastní algoritmus, který je už z velké části hotov.

JaVaFo je spustitelný jako samostatný jar program. Na vstup dostane soubor v TRF formátu. Tournament Report File (TRF) definuje na kterých pozicích mají být jaké údaje. Ukázka TRF formátu pro hráče je zobazena v tabulce 5.1. Výstup je textový soubor, kde na první řádce je počet párů a na každé další řádce jsou startovní čísla hráčů, kteří spolu budou dané kolo hrát.

Při ukončení registrace hráčů si tedy potřebné informace o turnaji a o registrovaných hráčích převedu do formátu TRF a uložím je. Po každém ukole daný TRF soubour přepíšu, aby odpovídal aktuálnímu stavu turnaje. Příkaz pro spuštění programu vypadá následovně:

```
1 rt.exec("java -ea -jar ./javafo.jar " + this.filePath + " -p " +
    ↪ this.outputPath);
```

Zdrojový kód 5.2: Volání programu JaVaFo

5.4 Round-robin systém

Při implementaci round-robin systému jsem postupovala podle algoritmu popsaného v kapitole 1.2.1. K ukládání starovních čísel hráčů jsem použila dvourozměrné pole, které reprezentuje Bergrovu tabulku pro n hráčů, kde každý řádek představuje jedno kolo. Pole je indexováno od nuly.

Nejdříve se vytvoří počáteční řádek tabulky, obsahující n hráčů, seřazených podle jejich startovních čísel od nejnižší po nejvyšší. Hráč na pozici i se utká s hráčem na pozici k, kde $k = n - i - 1$. Hráči na pozicích i a k se utkají na šachovnici číslo $i + 1$, kde hráč na pozici i bude mít bílou a hráč na pozici k černou barvu. Výjimku při přidělování barev tvoří hráči na první šachovnici u nichž se barva určuje podle toho jakou měl naposled hráč, na indexu k, barvu. Tento hráč začíná vždy černou barvou a následně se mu barvy vždy mění.

Pro každé další kolo se každý hráč posune o $n/2$ pozic, pouze hráč se startovním číslem n se neposune. Tedy hráč na aktuální pozici i skončí na pozici $(i + n/2) \pmod{n - 1}$.

5.5 Průběžné pořadí hráčů

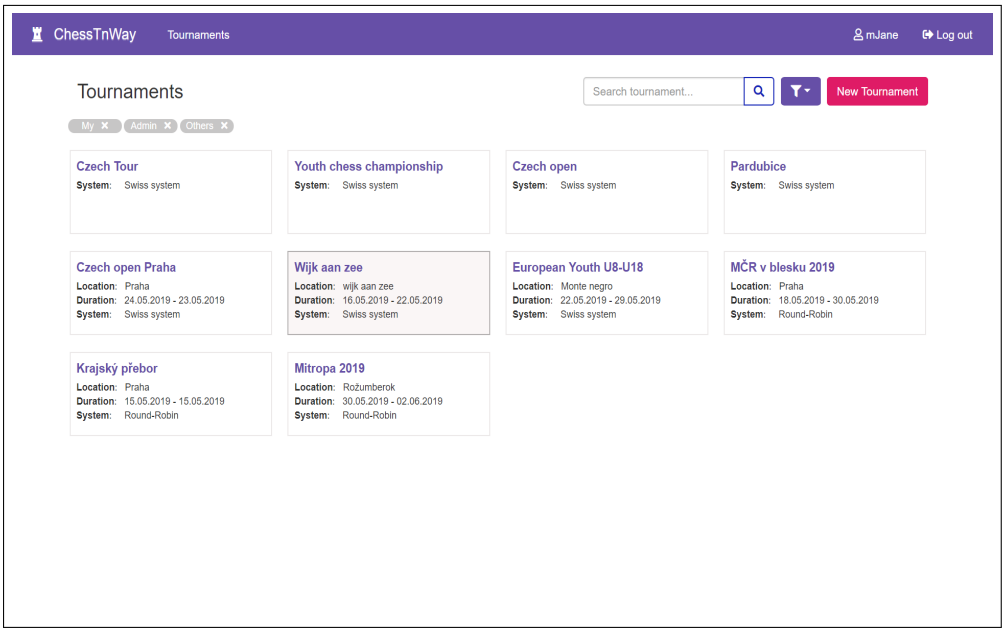
Jeden z problémů, které jsem při implementaci řešila, bylo průběžné řazení hráčů. Řadící algortimus musí být snadno přizpůsobitelný, protoženení předem dané podle jakých pomocných hodnocení se hráči budou po jednotlivých kolech řadit.

Tento problém jsem řešila tak, že jsem vytvořila `enum TieBreakComparator`, který implementuje `Comparator`. Každý prvek ve výpočtovém typu představuje pomocné hodnocení a obsahuje funkci `compare`, která v závislosti na daném enumu porovnává pomocné hodnocení. Tento výpočotvý typ obsahuje funkci `getComparator`, která dostane jako parametr list `TieBreakComparator` a vrací `Comparator`.

5.6 Adapter pattern

Adapter pattern se používá pokud je potřeba nějkému objektu přidat další vlastnost a při tom nezměnit jeho chování.

V mém případě jsem tento pattern využila při zobrazení detailu turnaje. Pokud je turnaj viditelný všem, může si jeho detail zobrazit i uživatel, který nemá administrátorské oprávnění k danému turnaji. Proto při dotazování se na turnaj z databáze zjistím i informaci, jaký má uživatel k turnaji práva a ty danému turnaji nastavím. Tedy z databáze vyberu turnaj typu `Tournament` z něho vytvořím typ `TournamentAdapter` pomocí funkce z patřičné servici mu nastavím `UserRole` a následně `TournamentAdapter` vracím klientovi.



Obrázek 5.1: Výpis turnajů

5.7 Uživatelský interface

Výslednou aplikaci jsem rozdělila do přibližně patnácti stránek. Níže jsou popsány některé z nich.

Přihlašovací stránka

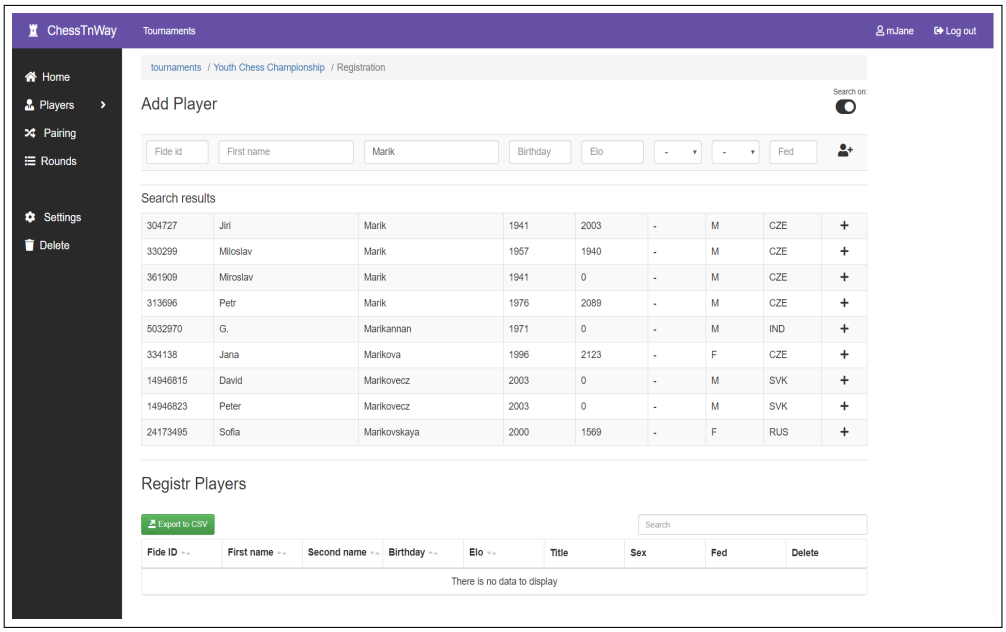
Cesta: `/login`
Na tuto stránku je uživatel přesměrován po zadání hlavní adresy aplikace `www.tnchessmanager.cz/`. Na stránce se nachází přihlašovací formulář. Dále má uživatel možnost přihlásit se jako host nebo se přesměrovat na registrační formulář.

Výpis turnajů

Cesta: `/tournaments`
Na tuto stránku je uživatel přesměrován po přihlášení nebo registraci. Stránka obsahuje výpis turnajů, možnost vyhledávání a filtraci turnajů a tlačítko na vytvoření nového turnaje, viz obrázek 5.1.

Vytvoření turnaje

Cesta: `/newTournament`
Formulář pro vytváření turnaje. Tento formulář se skládá celkem ze tří kroků. Po vytvoření turnaje je uživatel přesměrován na jeho domovskou stránku.



Obrázek 5.2: Registrace hráčů

Registrace hráčů

Cesta: `/tournament/:tournamentId/tnPlayers`
Na této stránce je formulář pro registraci uživatelů a výpis již registrovaných uživatelů. Uživatel má k dispozici tlačíkto, kterým může zapnou a vypnou vyhledávání v databázi reálných hráčů.

Výpis kola

Cesta: `/tournament/:tournamentId/games/:roundId`
Na této stránce jsou zobrazeny partie k určitému kolu. Pokud se jedná o aktuální kolo má oprávněný uživatel k dispozici formulář na zadávání výsledků a tlačítko na ukončení kola. Uživatel se může mezi jednotlivými koly turnaje navigovat pomocí tlačítek *Prev* a *Next*.

Profil hráče

Cesta: `/tournament/:tournamentId/tnPlayer/:tnPlayerId`
Na této stránce jsou umístěny údaje o hráči, počet bodů a tabulka, ve které je vyznačeno s kým jaké kolo hrál, jakou měl barvu a výsledek zápasu.

Kapitola 6

Testování

V této kapitole jsou popsány způsoby testování aplikace a to se zaměřěním na funkční testy, unit testy a uživatelské testování.

6.1 Funkční testy

Funkční testy mají za úkol prověřit, zda jsou implementovány veškeré definované funkční požadavky. V mém případě jsem kontrolovala zda nasazený program umožňuje veškeré funkční požadavky definované v kapitole 3.1.1. Při tomto testování jsem postupovala podle případů užití z kapitoly 3.3.

Díky tomuto testování jsem odhalila některé chyby, jako například, že turnaje mohl mazat i uživatel bez dostatečného oprávnění. Všechny tyto chyby byly opraveny, aby vyhovovaly funkčním požadavkům a případům užití.

6.2 Unit testy

Termín *unit testy* pochází z doby před érou objektového programování a odkazuje, že testy slouží k testování jednotlivých *unit*, tedy kusů systému. V dnešní době se též používá termín *component tests*. [38] Unit testy fungují jako separátní programy, které mají otestovat správnost činnosti funkcí.

Tyto testy se píší v průběhu vývoje softwaru a bývá zvykem, že se nejdříve napíše test a až poté daná funkce. Pro testování serveru jsem použila technologie Mockito a JUnit. JUnit je framework pro automatické testování unit testů [38]. Testovala jsem především service, zda splňují požadovanou funkcionalitu. Unit Testy na serveru mi pomohly odhalit chyby při změnách kódu.

Příklad testovací třídy, která testuje třídu `PlayerService` je zobrazen na zdrojovém kódu 6.1. Unit test pak představuje funkce `getPlayerByName()`. Tento test slouží k prověření zda se po zavolání funkce `getPlayersByName()` ve třídě `PlayerService` provolá správná funkce v `repository` se správnými parametry.

```
1  @RunWith(MockitoJUnitRunner.class)
2  public class PlayerServiceTest {
3
4      @InjectMocks
5      private PlayerService playerService;
6
7      @Mock
8      private PlayerRepository playerRepository;
9
10     @Test
11     public void getPlayersByName() {
12         String playerName = "Jan";
13         playerService.getPlayersByName(playerName);
14         verify(playerRepository, times(1)).findAllByNamePrefix(
               ↪ playerName + "_%");
15     }
16    ...
17 }
```

Zdrojový kód 6.1: Příklad unit testu

6.3 Uživatelské testování

Uživatelské testování slouží především ke zjištění míry uživatelské přívětivosti aplikace [39].

6.3.1 Testovací uživatelé

Pro účely testování aplikace jsem oslovila pět uživatelů, kteří simulovali organizování turnaje. Právě pět uživatelů jsem zvolila protože podle [40] se jedná o ideální počet uživatelů, kteří jsou schopni odhalit problematické části aplikace.

Protože mou snahou bylo vytvořit program, který bude intuitivní, zvolila jsem tři testovací užovatele, kteří neměli předchozí zkušenosti s žádným z existujích programů na losování šachových turnajů. Tito uživatelé mají dlouholetou šachovou zkušenost a účastnili se mnoha šachových turnajů, vždy z pozice hráče.

Jeden z testovacích uživatelů se zúčatnil mnoha šachových turnajů několikrát i z pozice rozhodčího. Od rozhodčího se očekává, že bude práci s losovácím programem ovládat. Tento uživatel používal převážně Swiss Manager. Snažil se pracovat s programem Vega, který se mu ale nepodařilo zprovoznit.

Poslední testovací uživatel se účastnil šachových turnajů převážně z pozice rozhodčího. Tento uživatel má zkušenosti s programem Swiss Manager.

6.3.2 Testovací scénáře

Pro účely uživatelského testování jsem sestavila dva scénáře, podle kterých měli uživatelé postupovat. U každého kroku je uvedeno, zda s ním měli testovací uživatelé nějaké problémy.

Scénář č.1 slouží k odhalení problematických kroků při vytváření turnaje se systémem round-robin a obsahuje následující kroky:

1. **Registrovat se**

   S tímto krokem neměl nikdo z uživatelů žádný problém. Uživateli č.4 nebylo zřejmé co znamená tlačítko *Login as quest* v přihlašovacím formuláři.

2. **Vytvořit turnaj se systémem round-robin**

   Při zadávání údajů turnaje nebylo uživateli č.1 zřejmé, jaké informace jsou povinné a jaké ne. Uživatelé č.1 a č.2 neznali některá pomocná hodnocení nebo si nepamatovali jejich význam.

3. **Změnit název turnaje**

   Se změnou názvu turnaje neměl žádný z uživatelů problémy.

4. **Přidat do turnaje hráče**

   S přidáváním hráčů pomocí formuláře na stránce *Registration* neměl žádný z testovacích uživatelů problémy. Nikdo z testovacích uživatelů nepřidával hráče z existující databáze hráčů. Až po upozornění nalezli tláčítko, které výsledky vyhledávání zobrazuje. Užital č.3 upozornil, že si nevšiml, že hráče do turnaje již přidal. Uživateli č. 1 nebylo zřejmé, že pro vyhledávání v reálné databázi je potřeba psát do formuláře.

5. **Ukončit registraci**

   Uživateli č.3 a č.2 nebylo zřejmé, kolik hráčů musí přidat, aby bylo možné ukončit registraci. Po ukočení registrace nebylo většině uživatelům jasné, že se nalosovala všechna kola turnaje. S navigací na aktuální kolo turnaje neměl nikdo s uživatelů problém.

6. **Zadat výsledky aktuálního kola**

   S tímto krokem neměl žádný z testovacích uživatelů problémy.

7. **Zadat výsledky následujícího kola**

   Tento krok tedy obsahoval i krok ukončit předchozí kolo. Uživatel č.1 se nejdříve navigoval na další stránku, tlačítka *End round* si všiml až když mu nešly zadat výsledky u následujícího kola.

8. **Zobrazit průběžné pořadí po 1. kole**

   Uživateli č.1 nebylo jasné, kde se nachází průběžná pořadí.

Scénář č.2 sloužil k otestování kroků při tvorbě turnaje švýcarským systémem. Tento scénář obsahuje následující kroky:

1. **Vytvořit turnaj švýcarským systémem**

   Uživatel č.3 se nejdříve přihlásil jako *host*. Po kliknutí na tlačitko *Create tournament* se uživatel přesměroval zpět na přihlašovací stánku a úspěšně se zaregistroval a přihlásil. Učivateli č.4 nebylo zřejmé co znamená formulář s možností zadat poznámku.

2. **Nastavit pomocná hodnocení**

   Uživatelé č.2 a č.3 neznali některá pomocná hodnocení. Uživateli č.1 nebylo zřejmé, že tabulka pod výběrem pomocného hodnocení reflektuje defaultně nastavná pomocná hodnocení.

3. **Nalosovat první kolo turnaje**

   Všichni uživatelé se po ukončení regisrace navigovali na stránku *Pairing*. Tento krok tedy proběhl bez problémů.

4. **Zadat výsledky k aktuálnímu kolu**

   S tímto krokem neměli testovací uživatelé žádné problémy.

5. **Zobrazit průběžné pořadí po 1. kole**

   S navigací na stránku neměl uživatel žádné potíže, na stránce ovšem nebyla žádná data.

6. **Zobrazit startovní listinu hráčů**

   Uživateli č.4 nebylo zřejmé na jaké stránce se startovní listina nachází.

7. **Zobrazit profil libovolného hráče**

   S tímto krokem neměl žádný z uživatelů problém.

8. **Nalosovat další kolo**

   Tento krok tedy opět zahrnoval ukončení předchozího kola a poté navigaci do záložky *Pairings*. Tento krok zvládli všichni uživatele bez problémů.

6.3.3 Shrnutí

Všem uživatelům dělalo největší problém přidat hráče z existující databáze hráčů. Proto jsem ke stránce přidala doplňující informace k tlačítku na vypínání a zapínání vyhledávání a také jsem nastavila, aby bylo vyhledávání defaultně zapnuto. Ve formuláři na vytváření turnajů jsem pak zvýraznila povinné údaje červeným znakem „*“, viz obrázek D.7.

Testovací uživatelé uvedli, že jim aplikace až na zmíněné připomínky přišla přehledná.

Kapitola 7

Možnosti rozšíření

V této kapitole jsou uvedeny požadavky, které mě při návrhu a implementaci aplikace napadly a které by bylo dobré realizovat v další verzi aplikace.

7.1 Licence FIDE

Aby byla aplikace rozšiřitelná i na mezinárodní úrovni, bylo by dobré splnit požadavky na zisk licence FIDE zmíněné v [41]. Licence FIDE se vždy vztahuje na konkrétní systém. Tyto požadavky obsahují:

- **L1: FIDE mód** – program musí implicitně být ve FIDE módu.
- **L2: Párování hráčů** – hráči musí být párování podle startovních čísel ne podle ratingu.
- **L3: Startovní čísla** – startovní čísla nemohou být změněna po 4. kole turnaje.
- **L4: Import/Export dat** – možnost importu a exportu dat v trf formátu.
- **L5: Zkušební verze** – uživatel musí mít možnost vyzkoušet si práci s programem zdarma minimálně v nějaké upravené verzi.
- **L6: Generování turnaje** – možnost generování simulovaného turnaje.
- **L7: Zrychlené systémy** – musí být implementován zrychlené systémy.
- **L8: Formát výsledků** – musí být možné zadávat specialní výsledky.
- **L9: Body za bye** – možnost definovat počet bodů za bye. Toto hodnocení se v průběhu turnaje nesmí měnit.
- **L10: Rating listina** – program musí poskytovat oficiální FIDE rating listinu.

Pokrytí požadavků na licenci FIDE je znázorněno tabulkou 7.1

L1	✗	
L2	✓	
L3	✓	
L4	✗	Převedení do formátu TRF je imlementováno.
L5	✓	
L6	✗	Generování turnajů je umožněno programem JaVaFo.
L7	✗	Zrychlené systémy jsou implementovány v JaVaFo.
L8	✓	
L9	✗	Body za bye se dají měnit v průběhu turnaje.
L10	✓	

Tabulka 7.1: Pokrytí požadavků na licenci FIDE

7.2 Funkční požadavky

FR1: Volby kategorií

U každého turnaje se vyhlašují jiné katogorie (například cena za nejmladšího hráče atd.), proto by bylo dobré, aby organizátoři mohli zadat, jaké se budou vyhlašovat.

FR2: Volby atributů u hráčů

Hráči obsahují mnoho atributů a záleží na turnaji, které z těchto atributů je potřeba. Například u turnaje, kde se vyhlašují věkové kategorie je dobré uvádět rok narození u hráčů. Proto by bylo dobré, kdyby organizátoři měli možnost výběru atributů, které se u hráčů budou uvádět.

FR3: Ruční párování hráčů

U turnajů, které nejsou určeny pro zápočet na elo, se někdy používá tzv. ruční párovaní. K tomuto párování se přistupuje v okamžiku, kdy se ví dopředu, že se někteří hráči nebudou moci dostavit a tak se napárují proti sobě.

FR4: Možnost pořádání týmových soutěží

Aby bylo možné pořádat týmové soutěže musí se doplnit pomocná hodnocení, která jsou pro ně určena (například součet bodů hráču v týmu). Dále je potřeba aby bylo možné zadat tým a dále jeho hráče, kterých může být ruzný počet.

FR5: Doplnění dalších švýcarských systémů

Pouze jeden ze zmíněných programů z kapitoly 2.1 a to Vega je schválený pro systém Dubov. Holadský systém je sice nejpoužívanější, na některých turnajích se nicméně volí jiné systémy. Například na šachové olympiádě se používá právě losovací systém Dubov.

7.3 Nefunkční požadavky

NFR1: Desktopová aplikace

K zabalení do desktopové aplikace slouží Electron. Electron je opensorce knihovna, kterou vyvinul GitHub.[43]

NFR2: Responzivita

V průběhu turnaje u sebe mají rozhodčí vždy počítač, pomocí kterého ovládají losovací program. Pro hráče je ale rozhodně příjemnější podívat se na rozlosování například na mobilu. Proto by bylo dobré v další verzi vyřešit respozivitu aplikace.

NFR3: Vícejazyčnost

Aktuálně je program pouze v anglickém jazyce. Do budoucna by bylo dobré, aby byl program také v češtině.

Závěr

Cílem této bakalářské práce bylo implementovat a nasadit systém na podporu organizace šachových turnajů. Součástí návrhnu byla analýza cílové skupiny uživatelů a existujících programů zabývajících se stejnou oblastí. Implementace pak měla proběhnout na základě návrhu a výběru vhodné technologie. Další z dílčích cílů bylo systém řádně otestovat, zhodnotit možnosti rozšíření a nasadit alespoň uzavřené skupině uživatelů.

Všechny tyto části jsem úspěšně splnila. Systém jsem vyvinula jako webovou aplikaci, která je nasazena na produkčním serveru. V práci jsem se zabývala dvěmi nejčastěji používanými systémy na losování turnajů a to švýcarským systémem a round-robin systémem.

Vyzkoušela jsem si práci s osmi již existujícími programy, většina z nich byla placená, ale nabízely zkušební verzi zdarma. Za účelem analýzy cílové skupiny uživatelů jsem vytvořila dotazník, který jsem distribuovala mezi organizátory turnajů.

Při implementaci aplikace jsem kladla důraz na volbu moderních technologií, aby byla aplikace lehce rozšiřitelná. Za tímto účelem byla vytvořena i dokumentace k serverové části projektu.

Během návrhu a implementace aplikace, mě napadlo mnoho vylepšení a doplnění požadavků, které byly již nad rámec dané práce. Tyto požadavky jsou sepsány v možnostech rozšíření a do budoucna bych ráda na vývoji aplikace pokračovala a dané požadavky implementovala.

Při analýze existujících řešení jsem byla překvapená nad stavem reálně používaných programů a domnívám se, že by aplikace při dobré propagaci mohla nalézt reálné využití.

Literatura

[1] Šachový svaz České republiky: *STANOVY Šachového svazu České republiky z. s.* [online]. 2018, [cit. 1. 4. 2019]. Dostupné z: https://www.chess.cz/wp-content/uploads/2018/02/Stanovy_SSCR_2018.pdf

[2] Handbook: General handling rules for Swiss Tournaments. *World Chess Federation* [online]. [cit. 2019-04-01]. Dostupné z: https://www.fide.com/fide/handbook.html?id=84&view=article

[3] Handbook: FIDE Rating Regulations effective from 1 July 2017. *World Chess Federation* [online]. 2017 [cit. 2019-04-20]. Dostupné z: https://www.fide.com/fide/handbook.html?id=197&view=article

[4] Top 100 Players April 2019 – Archive. *World Chess Federation* [online]. 2019 [cit. 2019-04-20]. Dostupné z: https://ratings.fide.com/toparc.phtml?cod=541

[5] General Regulations for Competitions. Annex 3: Tie-Break Regulations. *World Chess Federation* [online]. [cit. 2019-04-20]. Dostupné z: https://www.fide.com/fide/handbook.html?id=187&view=article

[6] FIDE: *General Regulations for Competitions* [online]. [cit. 4. 4. 2019]. Dostupné z: https://www.fide.com/FIDE/handbook/Competition_Rules.pdf

[7] Lucas, Edouard (1883).*Récréations Mathématiques* (in French). Paris: Gauthier-Villars. pp. 161–197.

[8] HEREJK, Petr. Pravidla pro turnaje hrané švýcarským systémem. *Šachový svaz České republiky* [online]. 2002 [cit. 2019-04-20]. Dostupné z: https://www.chess.cz/sachovy-svaz-cr/legislativa/pravidla-pro-turnaje-hrane-svycarskym-systemem/

[9] HELD Mario, *Tournament development with the Swiss Dutch System* [online]. [cit. 10. 4. 2019]. Dostupné z: http://arbiters.fide.com/various-contributions/487-dutch-system.html

[10] FIDE: *Endorsed Tournament Managers* [online]. [cit. 2019-04-03]. Dostupné z: https://www.fide.com/FIDE/handbook/C04Annex3_FEP18.pdf

[11] RICCA, Roberto. *JaVaFo, a pairing engine for the FIDE (Dutch) System* [online]. [cit. 2019-05-02]. Dostupné z: http://www.rrweb.org/javafo/JaVaFo.htm

[12] Nové licenční kódy pro program Swissmanager. *Šachový svaz České republiky z. s.* [online]. 2011 [cit. 2019-04-12]. Dostupné z: http://www.old.chess.cz/www/informace/komise/kr/dokumenty/nove-licencni-kody-pro-program-sm.html

[13] HERZOG, Heinz. An overview of important functions. *Swiss Manager* [online]. [cit. 2019-04-04]. Dostupné z: http://www.swiss-manager.at/

[14] *Vega: Chess Pairing Program* [online]. 2019 [cit. 2019-04-20]. Dostupné z: www.vegachess.com/ns/

[15] *Info64.org* [online]. [cit. 2019-04-12]. Dostupné z: https://info64.org/

[16] Looking for online swiss pairing software V: *Stack Exchange* [online]. 2017, [cit. 11. 4. 2019]. Dostupné z: https://www.chess.stackexchange.com/questions/8608/looking-for-online-swiss-pairing-software

[17] *SWIPS* [online]. [cit. 2019-04-10]. Dostupné z: www.chess.swips.eu

[18] Pricing. *SWIPS* [online]. [cit. 2019-04-10]. Dostupné z: https://chess.swips.eu/en/pricing

[19] *SwissSystem.org: Tournament management* [online]. 2018 [cit. 2019-04-20]. Dostupné z: www.swisssystem.org

[20] Functional and Nonfunctional Requirements: Specification and Types. *Alexsoft* [online]. [cit. 2019-04-20]. Dostupné z: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/

[21] Formát CSV. *Otevřená data* [online]. [cit. 2019-04-24]. Dostupné z: https://opendata.gov.cz/standardy:csv

[22] CHUNG, Lawrence, et al. Non-functional requirements in software engineering. Springer Science & Business Media, 2012.

[23] Jacobson, I.; Christerson, M.; Jonsson, P.; et al. Use-Case 2.0, The Guide to Succeeding with Use Cases. Addison-Wesley Professional, 1992, ISBN 978-0201544350.

[24] COCKBURN, Alistair. *Use cases: jak efektivně modelovat aplikace*. Brno: CP Books, 2005. ISBN 80-251-0721-3.

[25] Use Cases [online]. [cit. 2019-04-17] Dostupné z: `http://www.usability.gov/how-to-andtools/methods/use-cases.html`

[26] V. Rafe, R. Rafeh, S. Azizi and M. R. Z. Miralvand, *Verification and Validation of Activity Diagrams Using Graph Transformation* 2009 International Conference on Computer Technology and Development, Kota Kinabalu, 2009, pp. 201-205. doi: 10.1109/ICCTD.2009.172 URL: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5359784&isnumber=5359673`

[27] Freepik. *Computer Tower free icon* [online]. In: . [cit. 2019-05-15]. Dostupné z: `https://www.flaticon.com/free-icon/computer-tower_17231#term=desktop%20computer&page=1&position=94`

[28] Icon Works. *Laptop free icon* [online]. In: . [cit. 2019-05-15]. Dostupné z: `https://www.flaticon.com/free-icon/computer-screen_63337#term=computer&page=1&position=11`

[29] HTML: Hypertext Markup Language. *MDN web dosc: mozilla* [online]. 2019 [cit. 2019-04-11]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/HTML`

[30] BANKS, Alex a Eve PORCELLO. *Learning React: functional web development with React and Redux*. Beijing: O'Reilly Media, 2017. ISBN 978-1-491-95462-1.

[31] *The Front-End Tooling Survey 2018 - Results* [online]. 2018 [cit. 2019-04-02]. Dostupné z: `https://ashleynolan.co.uk/blog/frontend-tooling-survey-2018-results`

[32] GACKENHEIMER, Cory. *Introduction to React*. Apress, 2015.

[33] BUGL Daniel, *Learning Redux*, United Kingdom: Packt, 2017.

[34] *Bootstrap* [online]. [cit. 2019-04-02]. Dostupné z: `https://getbootstrap.com/`

[35] PUTANO, Ben. Most Popular and Influential Programming Languages of 2018. *Stackify* [online]. 2017 [cit. 2019-04-02]. Dostupné z: `https://stackify.com/popular-programming-languages-2018/`

[36] RAMAN, Raja CSP; DEWAILLY, Ludovic. *Building RESTful Web Services with Spring 5: Leverage the Power of Spring 5.0, Java SE 9, and Spring Boot 2.0.* Packt Publishing Ltd, 2018.

[37] SHIFF, Laura a Walker ROWE. NoSQL vs SQL: Examining The Differences and Deciding Which To Choose. *BMC Blogs* [online]. 2018 [cit. 2019-04-15]. Dostupné z: `https://www.bmc.com/blogs/sql-vs-nosql/`

[38] LINK, Johannes. *Unit testing in Java: how tests drive the code.* Elsevier, 2003.

[39] KRUG, Steve. *Rocket surgery made easy: The do-it-yourself guide to finding and fixing usability problems.* New Riders, 2009.

[40] NIELSEN, Jakob. *How Many Test Users in a Usability Study?* [online]. 2012, [cit. 2019-04-15]. Dostupné z: `https://www.nngroup.com/articles/how-many-test-users/`

[41] Handbook: C.04.A Appendix: Endorsement of a software program. *FIDE: World Chess Federation* [online]. [cit. 2019-04-11]. Dostupné z: `www.fide.com/fide/handbook.html?id=206&view=article`

[42] Download FRL April 2019. *FIDE: World Chess Federation* [online]. 2019 [cit. 2019-04-17]. Dostupné z: https://ratings.fide.com/download.phtml `https://ratings.fide.com/download.phtml`

[43] Electron Documentation. *Electron* [online]. [cit. 2019-04-17]. Dostupné z: `https://electronjs.org/docs/tutorial/about`

Příloha **A**

Seznam použitých zkratek

CM Candidat Master

CRUD Create, Read, Update, Delete

CSV Comma Separated Values

CSS Cascading Style Sheets

ES5 ECMAScript 5

FIDE Fédération Internationale des Échecs

FM Fide Master

GM Grand Master

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IM International Master

JAR Java ARchive

Java EE Java Enterprise Edition

JDBC Java Database Connectivity

JDK Java Development Kit

JPA Java Persistence Api

JSON JavaScript Object Notation

JVM Java Virtual Machine

MVC Model-View-Controller

ORM Object-Relational Mapping

REST Representational State Transfer

SQL Structured Query Language

ŠSČR Šachový svaz České republiky

TRF Tournament Report File

UML Unified Modeling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

WCM Woman Candidat Master

WFM Woman Fide Master

WGM Woman Grand Master

WIM Woman International Master

XML Extensible Markup Language

Příloha **B**

Begrovy tabulky

3&4 hráči
1. 1:4 2:3
2. 4:3 1:2
3. 2:4 3:1

5&6 hráčů
1. 1:4 2:5 3:4
2. 6:4 5:3 1:2
3. 2:6 3:1 4:5
4. 6:5 1:4 2:3
5. 3:6 4:2 5:1

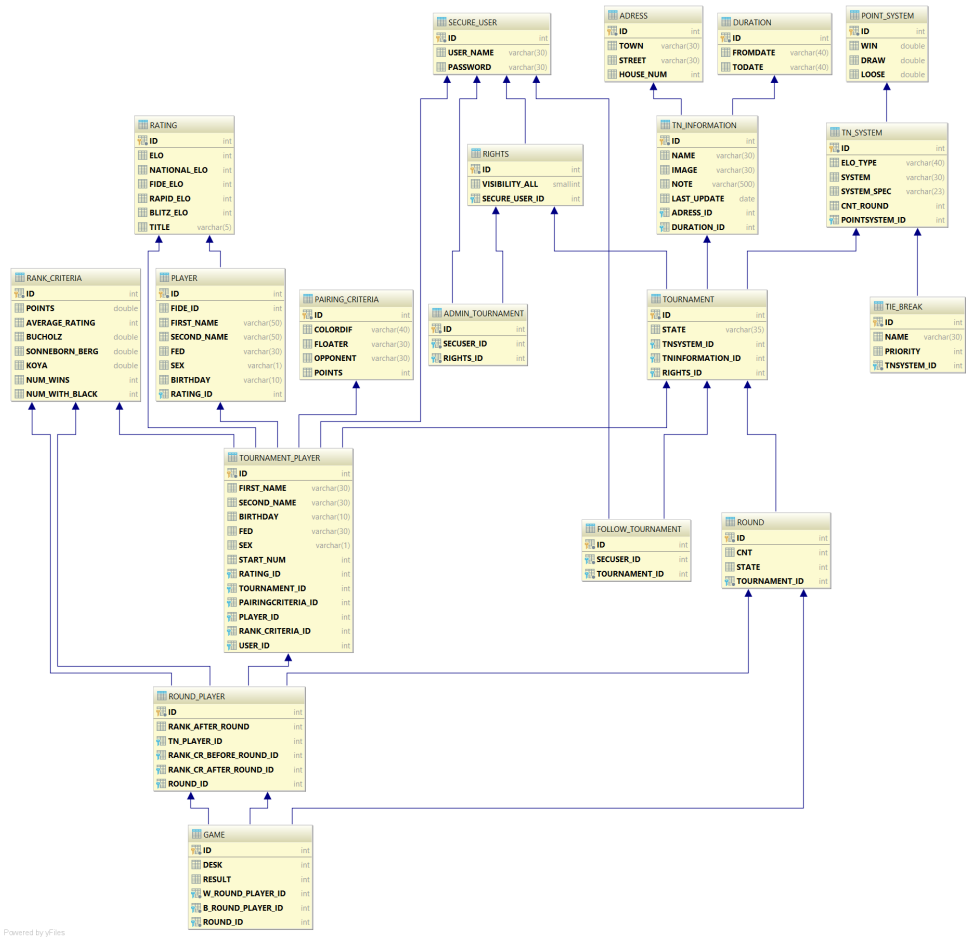
7&8 hráčů
1. 1:8 2:7 3:6 4:5
2. 8:5 6:4 7:3 1:2
3. 2:8 3:1 4:7 5:6
4. 8:6 7:5 1:4 2:3
5. 3:8 4:2 5:1 6:7
6. 8:7 1:6 2:5 3:4
7. 4:8 5:3 6:2 7:1

9&10 hráčů
1. 1:10 2:9 3:8 4:7 5:6
2. 10:6 7:5 8:4 9:3 1:2
3. 2:10 3:1 4:9 5:8 6:7
4. 10:7 8:6 9:5 1:4 2:3
5. 3:10 4:2 5:1 6:9 7:8
6. 10:8 9:7 1:6 2:5 3:4
7. 4:10 5:3 6:2 7:1 8:9
8. 10:9 1:8 2:7 3:6 4:5
9. 5:10 6:4 7:3 8:2 9:1

11&12 hráčů
1. 1:12 2:11 3:10 4:9 5:8 6:7
2. 12:7 8:6 9:5 10:4 11:3 1:2
3. 2:12 3:1 4:11 5:10 6:9 7:8
4. 12:8 9:7 10:6 11:5 1:4 2:3
5. 3:12 4:2 5:1 6:11 7:10 8:9
6. 12:9 10:8 11:7 1:6 2:5 3:4
7. 4:12 5:3 6:2 7:1 8:11 9:10
8. 12:10 11:9 1:8 2:7 3:6 4:5
9. 5:12 6:4 7:3 8:2 9:1 10:12
10. 12:11 1:10 2:9 3:8 4:7 5:6
11. 6:12 7:5 8:4 9:3 10:2 11:1

Příloha C

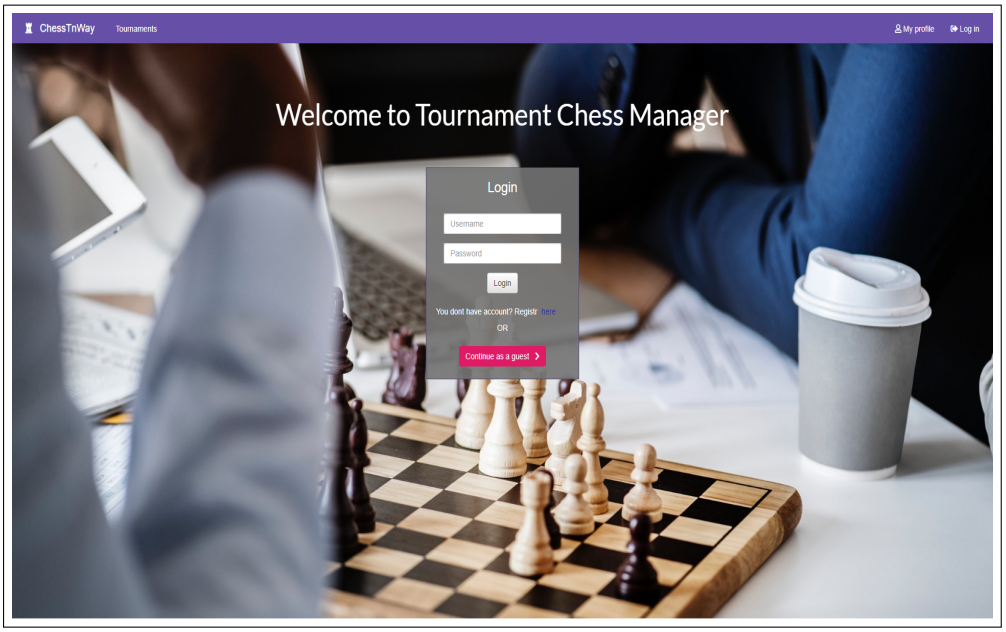
Databázový model



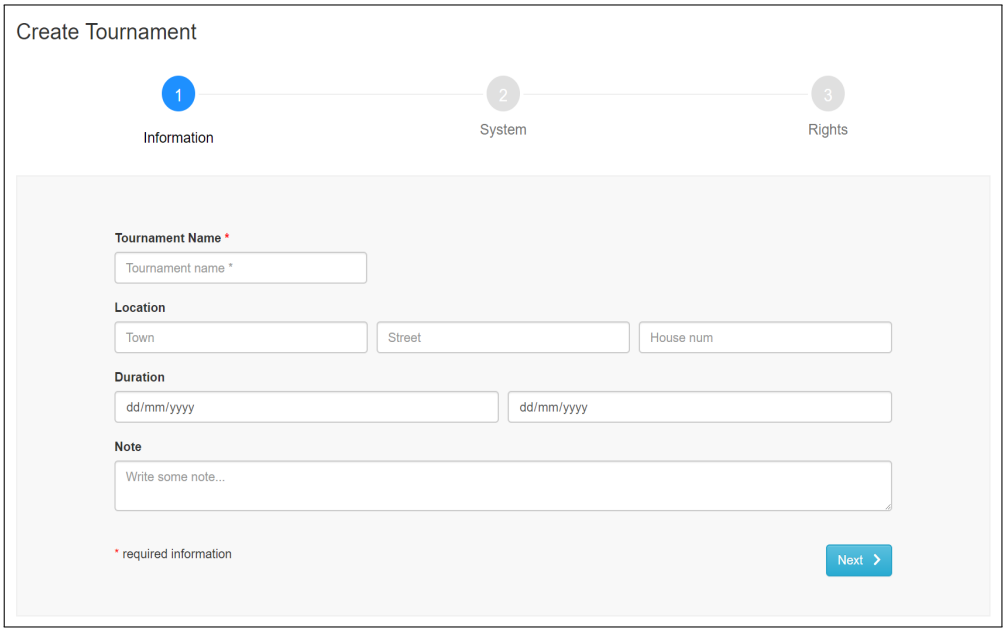
Obrázek C.1: Databázový model

Příloha D

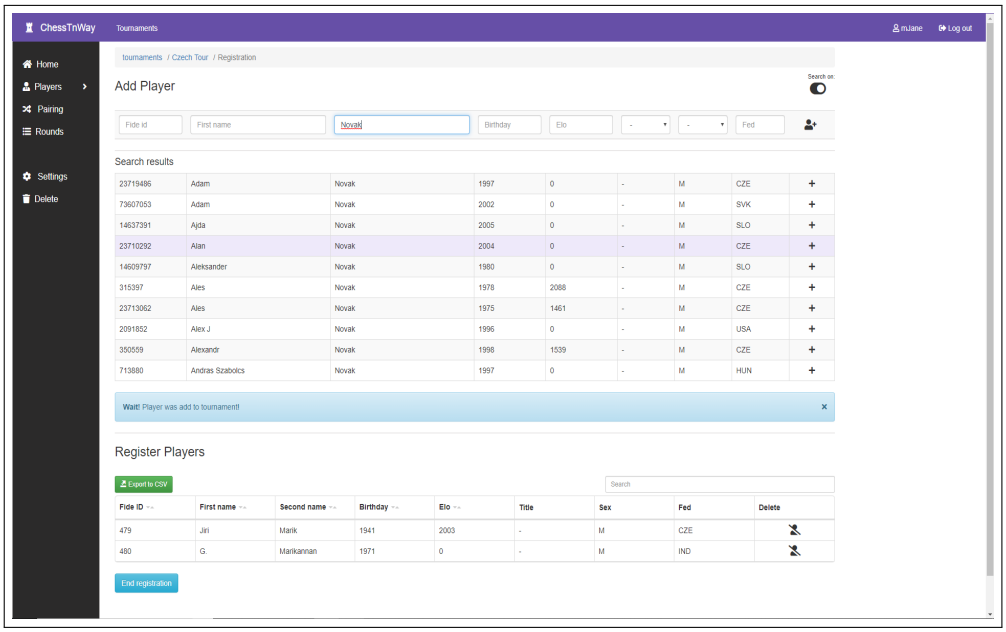
Výsledná aplikace



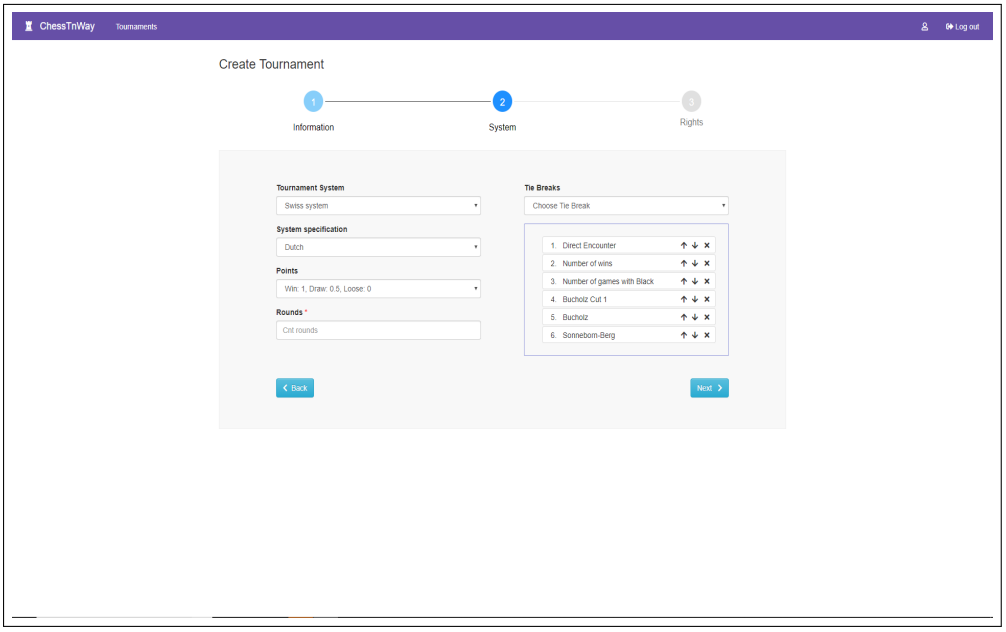
Obrázek D.1: Přihlašovací stránka



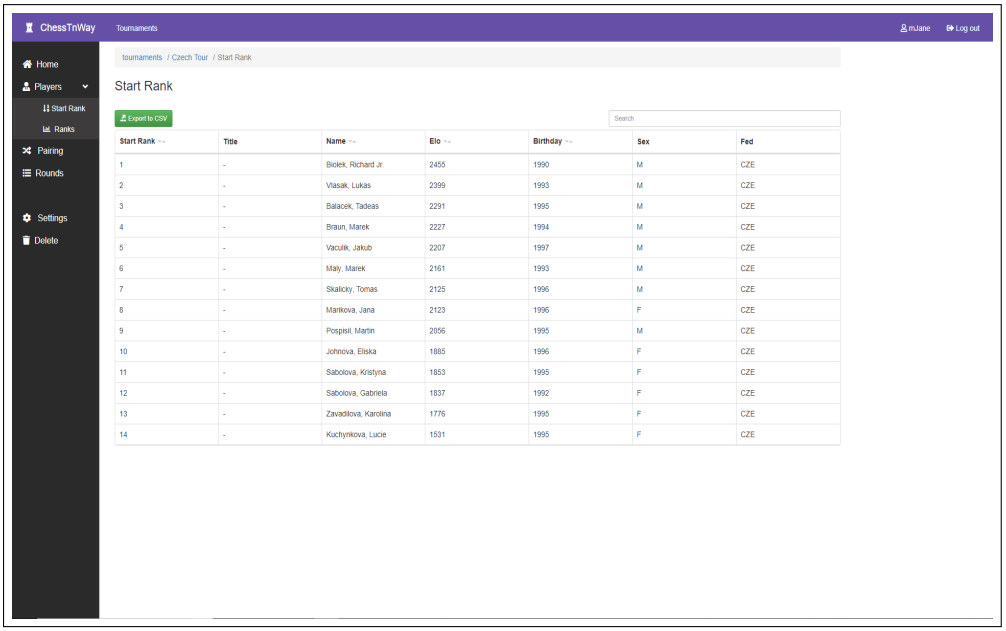
Obrázek D.2: Formulář vytváření turnaje



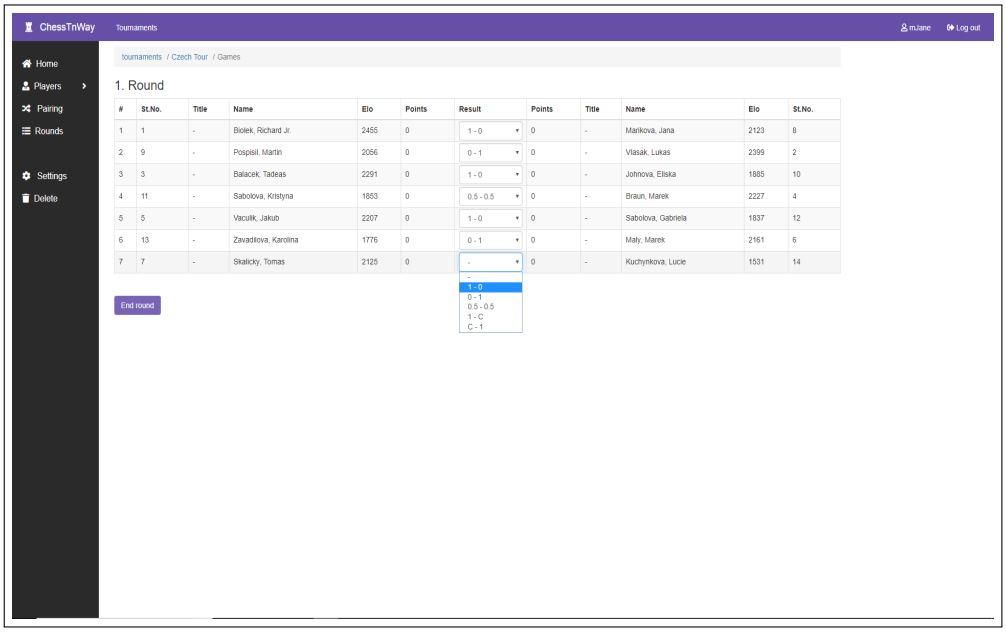
Obrázek D.3: Přidání hráče do turnaje



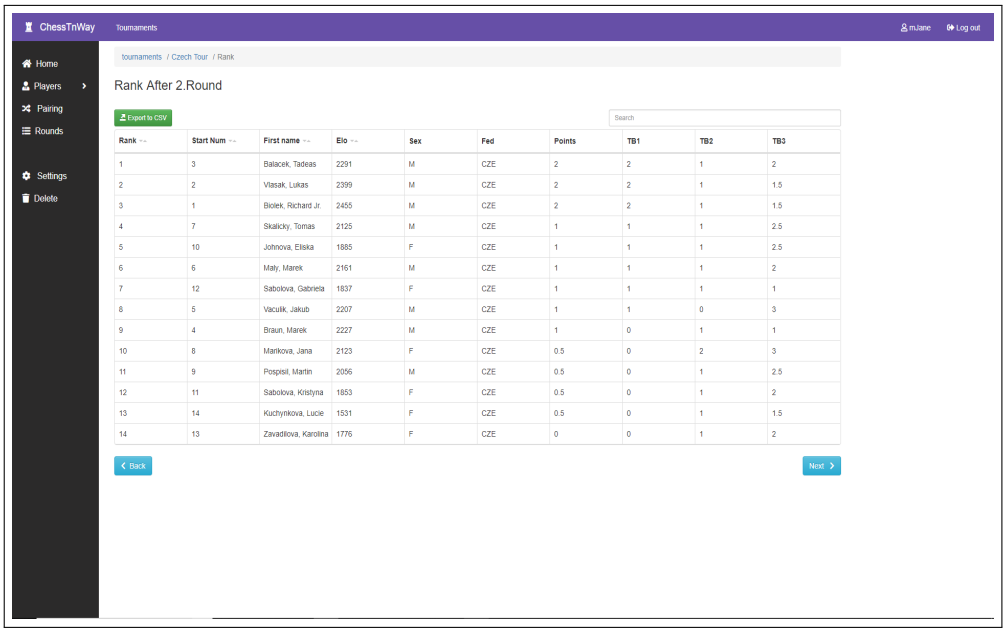
Obrázek D.4: Volba systému turnaje



Obrázek D.5: Startovní listina hráčů



Obrázek D.6: Zadávání výsledků



Obrázek D.7: Průběžné pořadí hráčů

Příloha **E**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `javadoc` ..... adresář s dokumentací
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `klient` ..... zdrojové kódy klienta
  - `server` ..... zdrojové kódy serveru
- `app`
  - `spusteni.txt` ..... příručka na spuštění
  - `tnManager.jar` ..... spustitelný soubor
  - `db` ..... scripty pro vytvoření databáze
- `text` ..... text práce
  - `BP_Marikova_Jana_2019.pdf` ..... text práce ve formátu PDF